REINFORCEMENT LEARNING AND STOCHASTIC OPTIMIZATION

A unified framework for sequential decisions

Warren B. Powell

August 22, 2021

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Optimization Under Uncertainty: A unified framework
Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

CHAPTER 9

MODELING SEQUENTIAL DECISION PROBLEMS

Perhaps one of the most important skills to develop when solving sequential decision problems is the ability to write down a mathematical model of the problem. As illustrated in figure 9.1, the path from a real application to doing computational work on the computer has to pass through the process of mathematical modeling. Unlike fields such as deterministic optimization and machine learning, there is not a standard modeling framework for decisions under uncertainty. This chapter will develop, in much greater detail, our universal modeling framework for any sequential decision problem. Although we have introduced this framework in earlier chapters, this chapter is dedicated to modeling, bringing out the incredible richness of sequential decision problems. This chapter is written to stand alone, so there is some repetition of elements of our universal model.

While the problem domain of sequential decision problems is astonishingly rich, we can write any sequential decision problem as the sequence:

$$(decision, information, decision, information, \ldots).$$

Let x_t be the decision we make at time t, and let W_{t+1} be the new information that arrives between t (that is, after the decision has been made), and $t+1$ (when we have to make the next decision). We will find it convenient to represent what we know when we make a decision. We refer to this information as the "state" variable S_t (think of this as our "state of knowledge"). Using this bit of notation, we can write our sequential decision problem as

$$(S_0, x_0, W_1, S_1, \ldots, S_t, x_t, W_{t+1}, S_{t+1}, \ldots, S_T). \tag{9.1}$$

Reinforcement Learning and Stochastic Optimization. By Warren B. Powell

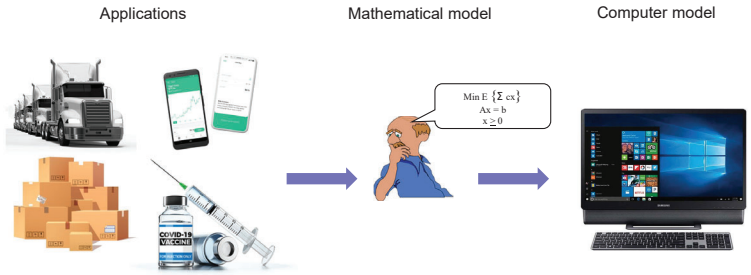


Figure 9.1 The path from applications to computation passes through the need for a mathematical model

There are many applications where it is more natural to use a counter n, which might be the n^{th} arrival of a customer, the n^{th} experiment, or the n^{th} iteration of an algorithm, in which case we would write our sequence as

$$(S^0, x^0, W^1, S^1, \ldots, S^n, x^n, W^{n+1}, S^{n+1}, \ldots, S^N, x^N). \tag{9.2}$$

Note that the n^{th} arrival might occur in continuous time, where we might let τ^n be the time at which the n^{th} event occurs. This notation allows us to model systems in continuous time (we can let t^n be the time of the n^{th} decision event).

There are problems where we might repeatedly simulate over time, in which case we would let write our sequence as

$$(S_0^1, x_0^1, W_1^1, \ldots, S_t^1, x_t^1, W_{t+1}^1, \ldots, S_0^n, x_0^n, W_1^n, \ldots, S_t^n, x_t^n, W_{t+1}^n, \ldots),$$

where we assume that our first pass is viewed as iteration $n = 1$. For the remainder of this chapter, we will assume our underlying physical process evolves over time, but any search for better policies (or parameters) will use iteration n.

After each decision, we evaluate our performance using a metric such as a contribution (there are many terms we might use) that we typically will write as $C(S_t, x_t)$ or, in some situations, $C(S_t, x_t, W_{t+1})$ (again, there are other styles that we discuss below). Decisions x_t are determined using a function we call a policy and denote by $X^\pi(S_t)$. Our ultimate goal is finding the best policy that optimizes the contribution function in some way.

The sequence in (9.1) (or (9.2) if we are using counts) can be used to describe virtually *any* sequential decision problem, but it requires that the problem be modeled correctly, and this means properly using one of the most misunderstood concepts in sequential decision problems: the state variable. The remainder of this chapter develops this basic model in considerably more depth.

Up to now, we have avoided discussing some important subtleties that arise in the modeling of sequential decision systems. We intentionally overlooked trying to define a state variable, which we have viewed as simply S_t. We have avoided discussions of how to properly model time or more complex information processes. We have also ignored the richness of modeling all the different sources of uncertainty for which we have a dedicated chapter (chapter 10). This style has facilitated introducing some basic ideas in dynamic programming, but would severely limit our ability to apply these methods to real problems.

There are five elements to any sequential decision problem, consisting of the following:

State variables - The state variables describe what we need to know (from history) to model the system forward in time. The initial state S_0 is also where we specify fixed parameters, the initial values of parameters (or quantities) that vary over time, as well as our distribution of belief about parameters we do not know perfectly.

Decision/action/control variables - These are the variables we control. Choosing these variables ("making decisions") represents the central challenge in sequential decision problems. This is where we describe constraints that limit what decisions we can make. Here is where we introduce the concept of a policy, but do not describe how to design the policy.

Exogenous information variables - These variables describe information that arrives to us exogenously, representing what we learn after we make each decision. Modeling exogenous information processes can be a significant challenge for many applications.

Transition function - This is the function that describes how each state variable evolves from one point in time to another. We may have explicit equations relating the next state to the current state, decision, and the exogenous information we learn after making the decision, for some, all, or none of the state variables.

Objective function - We assume we are trying to maximize or minimize some metric that is specified. This function describes how well we are doing at a point in time, and represents the foundation for evaluating policies.

An important point to make about the modeling framework is that there will at all times be a direct relationship to the software implementation. The mathematical model can be translated directly into software, and it will be possible to translate changes in the software back to the mathematical model.

We are going to start by illustrating these elements in the context of a simple energy storage problem in section 9.1. This is a nice starter problem because the state variable is fairly obvious. However, in section 9.9, we demonstrate how simple problems become complicated quickly using extensions of the initial energy storage application in section 9.1. The variations in section 9.9 introduces modeling issues that have never been addressed in the academic literature.

A reader can actually skip the entire rest of the chapter after this illustration if they are new to the field and just getting started. The remainder of the chapter will lay the foundation for modeling (and then solving) an exceptionally wide range of problems, including all the application areas covered by the fields presented in chapter 2, and all the applications sketched in chapter 8. We demonstrate these concepts when we show how to model the energy storage variations in section 9.9.

Even for more determined readers who are willing to read past section 9.1, we have still marked a number of sections with * that can be skipped on a first read.

This entire chapter is predicated on our "model first, then solve" style because we are going to present the model without addressing how we might solve it (hint: we will use one or more of our four classes of policies). This contrasts with the standard style used in the literature on sequential decision problems which is to present a method (see, for example, the introduction to reinforcement learning in section 2.1.6), but we also include so-called models where it is clear that the next step is to use Bellman's equation.

The rest of this chapter is organized as follows. We begin by describing the principles of good notation in section 9.2, followed by section 9.3 which addresses the subtleties of modeling time. These two sections lay the critical foundation for notation that is used

throughout the book. Notation is not as critical for simple problems, as long as it is precise and consistent. But what seems like benign notational decisions for a simple problem can cause unnecessary difficulties, possibly producing a model that simply does not capture the real problem, or making the model completely intractable.

The five elements of a dynamic model are covered in the following sections:

- State variables - section 9.4.
- Decision variables - section 9.5
- Exogenous information variables - section 9.6
- Transition function - section 9.7
- Objective function - section 9.8

We then present a more complex energy storage problem in section 9.9.

Having laid this foundation, we transition to a series of topics that can be skipped on a first pass, but which help to expand the readers appreciation of modeling of dynamic systems. These include:

Base models vs. lookahead models - Section 9.10 introduces the concept of base models (which is what we describe in this chapter) and lookahead models, which represent one of our classes of policies (described in much greater detail in chapter 19).

Problem classification - Section 9.11 describes four fundamental problem classes differentiated based on whether we have state-independent or state-dependent problems, and whether we are working in an offline setting (maximizing the final reward) or an online setting (maximizing cumulative reward).

Policy evaluation - Section 9.12 describes how to evaluate a policy using Monte Carlo simulation. This can actually be somewhat subtle. We have found that a good test of whether you understand an expectation is that you know how to estimate it using Monte Carlo simulation.

Advanced probabilistic modeling concepts - For readers who enjoy bridging to more advanced concepts in probability theory, section 9.13 provides an introduction to the vocabulary of measure-theoretic concepts and probability modeling. This discussion is designed for readers who do not have any formal training in this area, but would like to understand some of the language (and concepts) that measure-theoretic probability brings to this area.

Once we have laid out the five core elements of the model, we still have two components that we deal in much greater depth in subsequent chapters:

Uncertainty modeling - Chapter 10 deals with the exceptionally rich area of modeling uncertainty, which enters our model through the initial state S_0, which can capture uncertainties about parameters and quantities, and the exogenous information process $W_1, \ldots, W_T$. We recommend that when providing the basic model of our problem, the discussion of "exogenous information" should be limited to just listing the variables, without delving into how we model the uncertainties.

Designing policies - Chapter 11 describes in more detail the four classes of policies, which are the topic of chapters 12-19. We believe strongly that the design of policies can only occur *after* we have developed our model.

This chapter describes modeling in considerable depth, and as a result it is quite long. Sections marked with a '*' can be skipped on a first read. The section on more advanced probabilistic modeling is marked with a '**' to indicate that this is more difficult material.

9.1 A SIMPLE MODELING ILLUSTRATION

We are going to first describe a simple energy storage problem in an unstructured way, then we are going to pull the problem together into the five dimensions of a sequential decision problem. We do this by beginning with a plain English narrative (which we recommend for any problem).

Narrative: We have a single battery connected to the grid, where we can either buy energy from the grid or sell it back to the grid. Electricity prices are highly volatile, and may jump from an average of around $20 per megawatt-hour (MWh) to over $1000 per MWh (in some areas of the country, prices can exceed $10,000 per MWh for brief periods). Prices change every 5 minutes, and we will make the assumption that we can observe the price and then decide if we want to buy or sell. We can only buy or sell for an entire 5-minute interval at a maximum rate of 10 kilowatts (0.01 megawatts). The capacity of our battery storage is 100 kilowatts, which means it may take 10 hours of continuous charging to charge an empty battery.

To model our problem we introduce the following notation:

$$
\begin{aligned}
x_t &= \text{The rate at which we buy from the grid to charge the battery } (x_t > 0) \\
&\quad \text{or or sell back to the grid to discharge the battery } (x_t < 0). \\
u &= \text{The maximum charge or discharge rate for the battery (the power} \\
&\quad \text{rating, which is 10 kwh).} \\
p_t &= \text{The price of electricity on the grid at time } t. \\
R_t &= \text{The charge level of the battery.} \\
R^{max} &= \text{The capacity of the battery.}
\end{aligned}
$$

Assume for the purposes of this simple model that the prices p_t are random and independent over time.

If the prices were known in advance (which makes this a deterministic problem), we might formulate the problem as a linear program using

$$\max_{x_0,\ldots,x_T} \sum_{t=0}^{T} -p_t x_t \tag{9.3}$$

subject to:

$$
\begin{aligned}
R_{t+1} &= R_t + x_t, \\
x_t &\leq u, \\
x_t &\leq R^{max} - R_t, \\
x_t &\geq 0.
\end{aligned}
$$

Now let's introduce the assumption that prices p_t are random, and independent across time. Our first step is to replace the deterministic decision x_t with a policy $X^\pi(S_t)$ that depends on the state (that is, what we know), which we need to define (we will do this in a minute).

Next, imagine that we are going to run a simulation. Assume that we have compiled from history a series of sample paths that we are going to index by the Greek letter ω. If we have 20 sample paths, think of having 20 values of ω where each ω implies a sequence of prices $p_0(\omega), p_1(\omega), \ldots, p_T(\omega)$. Let Ω be the entire set of sample paths (you can think of this as the numbers $1, 2, \ldots, 20$ if we have 20 sample paths of prices). We assume each sample path occurs with equal likelihood.

We haven't yet described how we are going to design our policy (that is for later), but let's say we have a policy. We can simulate the policy for sample path ω and get the sample value of the policy using

$$F^{\pi}(\omega) = \sum_{t=0}^{T} -p_t(\omega) X^{\pi}(S_t(\omega)),$$

where the notation $S_t(\omega)$ for our as-yet undefined state variable indicates that, as we would expect, depends on the sample path ω (technically, it also depends on the policy π that we have been following). Next, we want to average over all the sample paths, so we compute an average

$$\bar{F}^{\pi} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} F^{\pi}(\omega).$$

This averaging is an approximation of an expectation. If we could enumerate every sample path, we could write

$$F^{\pi} = \mathbb{E} \sum_{t=0}^{T} -p_t X^{\pi}(S_t). \tag{9.4}$$

Here, we drop the dependence on ω, but need to remember that prices p_t, as well as the state variables S_t, are random variables because they do depend on the sample path. There will be many times that we will write the objective function using the expectation as in equation (9.4). Whenever you see this, keep in mind that we are virtually always assuming that we will approximate the expectation using an average as in equation (9.4).

Our final step is to find the best policy. We would write this objective using

$$\max_{\pi} \mathbb{E} \sum_{t=0}^{T} -p_t X^{\pi}(S_t). \tag{9.5}$$

So, we now have our objective function (equation (9.5)), which is frustratingly stated in terms of optimizing over policies, but we have yet to provide any indication how we do this! This is what we call "model first, then solve." In this chapter, we will only get as far as the objective function. This parallels what is done in every single paper on deterministic optimization or optimal control, as well as any paper on machine learning. They all present a model (which includes an objective function) and only then do they set about solving it (which in our setting means coming up with a policy).

At this point, we have two tasks remaining:

1) Uncertainty quantification We need to develop a model of any sources of uncertainty such as our price process. In a real problem, we would not be able to assume that the prices are independent. In fact, this is a fairly difficult problem.

2) Designing policies We need to design effective policies for buying and selling that depend only on the information in the state variable S_t.

Now that we have described the problem, presented our notation, and described the two remaining tasks, we are going to step back and describe the problem in terms of the five elements described above:

State variables - This has to capture all the information we need at time t. We can see that we need to know how much is stored in the battery R_t, and the grid price p_t, so

$$S_t = (R_t, p_t).$$

Decision variables - Clearly this is x_t. We then need to express the constraints on x_t which are given by

$$\begin{aligned} x_t &\leq u, \\ x_t &\leq R^{max} - R_t, \\ x_t &\geq 0. \end{aligned}$$

Finally, when we define the decision variables we also introduce the policy $X^\pi(S_t)$ to be designed later. We introduce it now because we need it to present the objective function. Note that we have defined the state variable S_t, so this is given.

Exogenous information variables - This is where we model any information that becomes available *after* we make the decision x_t. For our simple problem, this would be the updated price, so

$$W_{t+1} = p_{t+1}.$$

Transition function - These are the equations that govern how the state variables evolve over time. We have two variables in the state variables. R_t evolves according to

$$R_{t+1} = R_t + x_t.$$

The price process evolves according to

$$p_{t+1} = W_{t+1}.$$

In other words, we just observe the next price rather than derive it from an equation. This is an example of "model free dynamic programming." There are many problems where we observe S_{t+1} rather than compute it; here we have an instance where we compute R_{t+1} but observe p_{t+1}. As an alternative model, we might assume that we model the *change* in the price using

$$\hat{p}_{t+1} = \text{the change in the price from } t \text{ to } t+1,$$

which means that $W_{t+1} = \hat{p}_{t+1}$ and our transition function becomes

$$p_{t+1} = p_t + \hat{p}_{t+1}.$$

Objective function - Our contribution function at time t is given by

$$C(S_t, x_t) = -p_t x_t,$$

where p_t is pulled from the state variable S_t. We would then write our objective function as

$$\max_{\pi} \mathbb{E} \sum_{t=0}^{T} C(S_t, X^{\pi}(S_t)). \tag{9.6}$$

Now we have modeled our problem using the five dimensions outlined earlier. This follows the same style used for deterministic optimization, but adapted for sequential decision problems. As problems become more complicated, the state variable will become more complex, as will the transition function (which requires an equation for each state variable). It would be nice if we could write a model from top to bottom, starting with state variables, but that is not how it works in practice. Modeling is iterative.

We repeat: this is a good stopping point for readers new to the field. Sections 9.2 on notational style and 9.3 on modeling time are useful to develop notational skills. The remainder of the chapter is primarily useful if you want a solid foundation for more complex problems (we give a hint of this in the expanded energy storage problem in section 9.9). In particular, section 9.4 is an in-depth introduction to the notion of state variables, which get complicated very quickly, even with relatively modest extensions of a simple problem. We demonstrate precisely this process in section 9.9 which presents a series of seemingly modest extensions of this energy problem, focusing on how to model the state variable as we add details to the problem.

9.2 NOTATIONAL STYLE

Good modeling begins with good notation. The choice of notation has to balance traditional style with the needs of a particular problem class. Notation is easier to learn if it is mnemonic (the letters look like what they mean) and compact (avoiding a profusion of symbols). Notation also helps to bridge communities. Notation is a language: the simpler the language, the easier it is to understand the problem.

As a start, it is useful to adopt notational conventions to simplify the style of our presentation. For this reason, we adopt the following notational conventions:

Variables - Variables are *always* a single letter. We would never use, for example, CH for "cost of holding inventory."

Modeling time - We always use t to represent a point in time, while we use τ to represent an interval over time. When we need to represent different points in time, we might use $t, t', \bar{t}, t^{max}$, and so on. Time is *always* represented as a subscript such as S_t.

Indexing time - If we are modeling activities in discrete time, then t is an index and should be put in the subscript. So x_t would be an activity at time t, with the vector $x = (x_0, x_1, \ldots, x_t, \ldots, x_T)$ giving us all the activities over time. When modeling problems in continuous time, it is more common to write t as an argument, as in $x(t)$. x_t is notationally more compact (try writing a complex equation full of variables written as $x(t)$ instead of x_t).

Indexing vectors - Vectors are almost always indexed in the subscript, as in x_{ij}. Since we use discrete time models throughout, an activity at time t can be viewed as an element of a vector. When there are multiple indices, they should be ordered from

outside in the general order over which they might be summed (think of the outermost index as the most detailed information). So, if x_{tij} is the flow from i to j at time t with cost c_{tij}, we might sum up the total cost using $\sum_t \sum_i \sum_j c_{tij} x_{tij}$. Dropping one or more indices creates a vector over the elements of the missing indices to the right. So, $x_t = (x_{tij})_{\forall i, \forall j}$ is the vector of all flows occurring at time t. Time, when present, is always the innermost index.

Temporal indexing of functions - A common notational error is to index a function by time t when in fact the function itself does not depend on time, but depends on inputs that do depend on time. For example, imagine that we have a stochastic price process where the state $S_t = p_t$ which is the price of the asset, and x_t is how much we sell $x_t > 0$ or buy $x_t < 0$. We might want to write our contribution as

$$C_t(S_t, x) = p_t x_t.$$

However, in this case the *function* does not depend on time t; it only depends on data $S_t = p_t$ that depends on time. So the proper way to write this would be

$$C(S_t, x) = p_t x_t.$$

Now imagine that our contribution function is given by

$$C_t(S_t, x_t) = \sum_{t'=t}^{t+H} p_{tt'} x_{tt'}.$$

Here, the *function* depends on time because the summation runs from t to $t + H$.

Flavors of variables - It is often the case that we need to indicate different flavors of variables, such as holding costs and order costs. These are always indicated as superscripts, where we might write c^h or c^{hold} as the holding cost. Note that while variables must be a single letter, superscripts may be words (although this should be used sparingly). We think of a variable like "c^h" as a single piece of notation. It is better to write c^h as the holding cost and c^p as the purchasing cost than to use h as the holding cost and p as the purchasing cost (the first approach uses a single letter c for cost, while the second approach uses up two letters - the roman alphabet is a scarce resource). Other ways of indicating flavors is hats ($\hat{x}$), bars ($\bar{x}$), tildes ($\tilde{x}$) and primes (x').

Iteration counters - There are problems where it is more natural to count events such as customer arrivals, experiments, observations, or iterations of an algorithm, rather than representing the actual time at which a decision is being made.

We place iteration counters in the superscript, since we view it as indicating the value of a single variable at iteration n, as opposed to the n^{th} element of a vector. So, x^n is our activity at iteration n, while x^{n+1} is the value of x at iteration $n + 1$. If we are using a descriptive superscript, we might write $x^{h,n}$ to represent x^h at iteration n. Sometimes algorithms require inner and outer iterations. In this case, we use n to index the outer iteration and m for the inner iteration.

While this will prove to be the most natural way to index iterations, there is potential for confusion where it may not be clear if the superscript n is an index (as we view it) or raising a variable to the n^{th} power. One notable exception to this convention is

indexing stepsizes which we first saw in chapter 5. If we write α^n, it looks like we are raising α to the n^{th} power, so we use α_n.

Sets are represented using capital letters in a calligraphic font, such as $\mathcal{X}, \mathcal{F}$ or $\mathcal{I}$. We generally use the lowercase roman letter as an element of a set, as in $x \in \mathcal{X}$ or $i \in \mathcal{I}$.

Exogenous information - Information that first becomes available (from outside the system) at time t is denoted using hats, for example, $\hat{D}_t$ or $\hat{p}_t$. Our only exception to this rule is W_t which is our generic notation for exogenous information (since W_t *always* refers to exogenous information, we do not use a hat).

Statistics - Statistics computed using exogenous information are generally indicated using bars, for example $\bar{x}_t$ or $\overline{V}_t$. Since these are functions of random variables, they are also random. We do not use hats, because we have reserved "hat" variables for exogenous information.

Index variables - Throughout, i, j, k, l, m and n are always scalar indices.

Superscripts/subscripts on superscripts/subscripts - As a general rule, avoid superscripts on superscripts (and so forth). For example, it is tempting to think of x_{b_t} as saying that x is a function of time t, when in fact this means it is a function of b which itself depends on time.

For example, x might be the number of clicks when the bid at time t is b_t, but what this notation is saying is that the number of clicks just depends on the bid, and not on time. If we want to capture the effect of both the bid and time, we have to write $x_{b,t}$.

Similarly, the notation F_{T^D} cannot be used as the forecast of the demand D at time T. To do this, you should write F_T^D. The notation F_{T^D} is just a forecast at a time $t = T^D$ that might correspond to the time, say, at which a demand occurs. But if you also write F_{T^p} where it just happens that $T^D = T^p$, you cannot refer to these as different forecasts because one is indexed by T^D while the other is indexed by T^p.

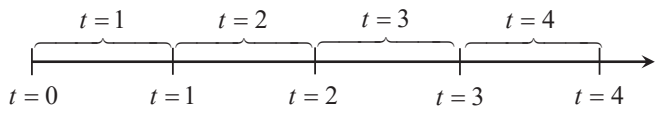
Of course, there are exceptions to every rule, and you have to keep an eye on standard notational conventions within pocket research communities.

9.3 MODELING TIME

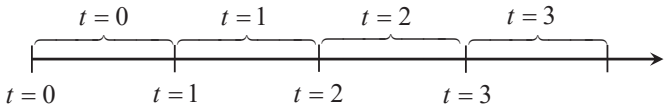
There are two strategies for modeling "time" in a sequential decision problem:

- Counters - There are many settings where we make decisions corresponding to discrete events, such as running an experiment, the arrival of a customer, or iterations of an algorithm. We generally let n the variable we use for counting, and we place it in the superscript, as in X^n or $\bar{f}^n(x)$. $n = 1$ corresponds to the first event, while $n = 0$ means no events have happened. However, our first decision occurs at $n = 0$ since we generally have to make a decision before anything has happened.

- Time - We may wish to directly model time. If time is continuous, we would write a function as $f(t)$, but all of the problems in this book are modeled in discrete time $t = 0, 1, 2, \ldots$. If we wish to model the time of the arrival of the n^{th} customer, we would write t^n. However, we would write X^n for a variable that depends on the n^{th} arrival rather than X_{t^n}.



9.2a: Information processes



9.2b: Physical processes

Figure 9.2 Relationship between discrete and continuous time for information processes (9.2a) and physical processes (9.2b).

Our style of indexing counters in the superscripts and time in subscripts helps when we are modeling simulations where we have to run a simulation multiple times. Thus, we might write X_t^n as information at time t in the n^{th} iteration of our simulation.

The confusion over modeling time arises in part because there are two processes that we have to capture: the flow of information, and the flow of physical and financial resources. For example, a buyer may purchase an option now (an information event) to buy a commodity in the future (the physical event). Customers may call an airline (the information event) to fly on a future flight (the physical event). An electric power company has to purchase equipment now to be used one or two years in the future. All of these problems represent examples of *lagged information processes* and force us to explicitly model the informational and physical events.

Notation can easily become confused when an author starts by writing down a deterministic model of a physical process, and then adds uncertainty. The problem arises because the proper convention for modeling time for information processes is different than what should be used for physical processes.

We begin by establishing the relationship between discrete and continuous time. All of the models in this book assume that decisions are made in discrete time (sometimes referred to as *decision epochs*). However, the flow of information is best viewed in continuous time.

The relationship of our discrete time approximation to the real flow of information and physical resources is depicted in figure 9.2. Above the line, "t" refers to a time interval while below the line, "t" refers to a point in time. When we are modeling information, time $t = 0$ is special; it represents "here and now" with the information that is available at the moment. The discrete time t refers to the time interval from $t - 1$ to t (illustrated in figure 9.2a). This means that the first new information arrives during time interval 1.

This notational style means that any variable indexed by t, say S_t or x_t, is assumed to have access to the information that arrived up to time t, which means up through time interval t. This property will dramatically simplify our notation in the future. For example, assume that f_t is our forecast of the demand for electricity. If $\hat{D}_t$ is the observed demand

during time interval t, we would write our updating equation for the forecast using

$$f_{t+1} = (1-\alpha)f_t + \alpha\hat{D}_{t+1}. \tag{9.7}$$

We refer to this form as the *informational representation*. Note that the forecast f_{t+1} is written as a function of the information that became available during time interval $(t, t+1)$, given by the demand $\hat{D}_{t+1}$.

When we are modeling a physical process, it is more natural to adopt a different convention (illustrated in figure 9.2b): discrete time t refers to the time interval between t and $t+1$. This convention arises because it is most natural in deterministic models to use time to represent when something is happening or when a resource can be used. For example, let R_t be our cash on hand that we can use during day t (implicitly, this means that we are measuring it at the beginning of the day). Let $\hat{D}_t$ be the demand for cash during the day, and let x_t represent additional cash that we have decided to add to our balance (to be used during day t). We can model our cash on hand using

$$R_{t+1} = R_t + x_t - \hat{D}_t. \tag{9.8}$$

We refer to this form as the *physical representation*. Note that the left-hand side is indexed by $t+1$, while all the quantities on the right-hand side are indexed by t.

Throughout this book, we are going to use the informational representation as indicated in equation (9.7). We first saw this in our presentation of stochastic gradients in chapter 5, when we wrote the updates from a stochastic gradient using

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

where here we are using iteration n instead of time t.

9.4 THE STATES OF OUR SYSTEM

The most important quantity in any sequential decision process is the state variable. This is the set of variables that captures everything that we know, and need to know, to model our system. Without question this is the most subtle, and poorly understood, dimension of modeling sequential decision problems.

9.4.1 Defining the state variable

Surprisingly, other presentations of dynamic programming spend little time defining a state variable. Bellman's seminal text [Bellman (1957), p. 81] says "... we have a physical system characterized at any stage by a small set of parameters, the *state variables*." In a much more modern treatment, Puterman first introduces a state variable by saying [Puterman (2005), p. 18] "At each decision epoch, the system occupies a *state*." In both cases, the italics are in the original manuscript, indicating that the term "state" is being introduced. In effect, both authors are saying that given a system, the state variable will be apparent from the context.

Interestingly, different communities appear to interpret state variables in slightly different ways. We adopt an interpretation that is fairly common in the control theory community, which effectively models the state variable S_t as all the information needed to model the system from time t onward. We agree with this definition, but it does not provide much

guidance in terms of actually translating real applications into a formal model. We suggest the following definitions:

Definition 9.4.1. *A* **state variable** *is:*

a) Policy-dependent version *A function of history that, combined with the exogenous information (and a policy), is necessary and sufficient to compute the cost/contribution function, the decision function (the policy), and any information required by the transition function to model the information needed for the cost/contribution and decision functions.*

b) Optimization version *A function of history that is necessary and sufficient to compute the cost/contribution function, the constraints, and any information required by the transition function to model the information needed for the cost/contribution function and the constraints.*

Some remarks are in order:

i) The policy-dependent definition defines the state variable in terms of the information needed to compute the core model information (cost/contribution function, and the policy (or decision function)), and any other information needed to model the evolution of the core information over time (that is, the transition function). Note that constraints (at a point in time t) are assumed to be captured by the policy. Since the policy can be any function, it could potentially be a function that includes information that does not seem relevant to the problem, and which would never be used in an optimal policy. For example, a policy that says "turn left if the sun is shining" with an objective to minimize travel time would put whether or not the sun is shining in the state variable, although this does not contribute to minimizing travel times.

ii) The optimization version defines a state variable in terms of the information needed to compute the core model information (costs/contributions and constraints), and any other information needed to model the evolution of the core information over time (their transition function). This definition limits the state variable to information needed by the optimization problem, and cannot include information that is irrelevant to the core model.

iii) Both definitions include any information that might be needed to compute the evolution of core model information, as well as information needed to model the evolution of this information over time. This includes information needed to represent the stochastic behavior, which includes distributional information needed to compute or approximate expectations. In section 9.9.4, we present an example of how rolling forecasts enter the state variable because they are needed in the transition function.

iv) Both definitions imply that the state variable includes the information needed to compute the transition function for core model information. For example, if we model a price process using

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon^p_{t+1}, \tag{9.9}$$

then the state variable for this price process would be $S_t = (p_t, p_{t-1}, p_{t-2})$. At time t, the prices p_{t-1} and p_{t-2} are not needed to compute the cost/contribution function or constraints, but they are needed to model the evolution of p_t, which is part of the cost/contribution function.

v) The qualifier "necessary and sufficient" is intended to eliminate irrelevant information. For example, with our lagged price model above, we need p_t, p_{t-1} and p_{t-2} but not p_{t-3} and earlier. A similar term used in the statistics literature is "sufficient statistic," which means it contains all the information needed for any future calculations.

vi) A byproduct of our definitions is the observation that *all* properly modeled dynamic systems are Markovian, by construction. It is surprisingly common for people to make a distinction between "Markovian" and "history-dependent" processes. For example, if our price process evolves according to equation (9.9), many would call this a history-dependent process, but consider what happens when we define

$$\bar{p}_t = \begin{pmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{pmatrix}$$

and let

$$\bar{\theta}_t = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{pmatrix}$$

which means we can write

$$p_{t+1} = \bar{\theta}^T \bar{p}_t + \varepsilon_{t+1}. \tag{9.10}$$

Here we see that $\bar{p}_t$ is a vector known at time t (who cares when the information first became known?). We would stay that equation (9.10) describes a Markov process with state $S_t = (p_t, p_{t-1}, p_{t-2})$.

vii) There is an issue of missing information and/or incorrect models. For example, we may assume that our price process evolves according to the model in equation (9.9), but this is really just an approximation of a much more complex process that is not known to us. As a simple illustration, assume that the true model is given by

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-1}^2 + \theta_3 p_{t-2} + \theta_4 p_{t-2}^2 + \theta_5 p_{t-1} p_{t-2} + \varepsilon_{t+1}^p. \tag{9.11}$$

We use equation (9.9) because it is simpler. Even if we tried equation (9.11), the noise in the data may lead us to conclude that θ_2, θ_4 and θ_5 are statistically indistinguishable from zero. If we had enough data, we might realize that the model (9.9) violates the assumptions that the error term ε_t is independent across time with the same distribution. If we knew that (9.11) was the true model (perhaps because we coded it into a simulator we are trying to optimize), we might say that the model in equation (9.9) is non-Markovian. For this issue, we turn to the famous quote by G.E.P. Box who noted: "All models are wrong, and some are useful," which is a way of saying there are errors in all models. The model in equation (9.9) is Markovian because we *assume* it to be Markovian.

There will be problems where we know that we do not know a parameter or quantity, but in these cases, the solution is to introduce a belief about these values. This belief is added to the state variable, which then produces a Markov model. If someone claims that a model is non-Markovian, then either it is missing known information that should be added, or we should add beliefs about unknown parameters and quantities.

These definitions provide a very quick test of the validity of a state variable. If there is a piece of data in either the decision function (policy), the transition function, or the contribution function which is not in the state variable, then we do not have a complete state variable. Similarly, if there is information in the state variable that is never needed in any of these three functions, then we can drop it and still have a valid state variable.

We use the term "necessary and sufficient" so that our state variable is as compact as possible. For example, we could argue that we need the entire history of events up to time t to model future dynamics, but in practice, this is rarely the case. As we start doing computational work, we are going to want S_t to be as compact as possible. Furthermore, there are many problems where we simply do not need to know the entire history. It might be enough to know the status of all our resources at time t (the resource variable R_t). But there are examples where this is not enough.

Assume, for example, that we need to use our history to forecast the price of a stock. Our history of prices is given by $(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_t)$. If we use a simple exponential smoothing model, our estimate of the mean price $\bar{p}_t$ can be computed using

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

where α is a stepsize satisfying $0 \leq \alpha \leq 1$. With this forecasting mechanism, we do not need to retain the history of prices, but rather only the latest estimate $\bar{p}_t$. As a result, $\bar{p}_t$ is called a *sufficient statistic*, which is a statistic that captures all relevant information needed to compute any additional statistics from new information. A state variable, according to our definition, is always a sufficient statistic.

Consider what happens when we switch from exponential smoothing to an N-period moving average. Our forecast of future prices is now given by

$$\bar{p}_t = \frac{1}{N}\sum_{\tau=0}^{N-1} \hat{p}_{t-\tau}.$$

Now, we have to retain the N-period rolling set of prices $(\hat{p}_t, \hat{p}_{t-1}, \ldots, \hat{p}_{t-N+1})$ in order to compute the price estimate in the next time period. With exponential smoothing, we could write

$$S_t = \bar{p}_t.$$

If we use the moving average, our state variable would be

$$S_t = (\hat{p}_t, \hat{p}_{t-1}, \ldots, \hat{p}_{t-N+1}). \tag{9.12}$$

Below we discuss latent variables (state variables that we *choose* to approximate as deterministic, but which really are changing stochastically over time), and unobservable state variables (which are also changing stochastically, but which we cannot observe).

9.4.2 The three states of our system

To set up our discussion, assume that we are interested in solving a relatively complex resource management problem, one that involves multiple (possibly many) different types of resources which can be modified in various ways (changing their attributes). For such a problem, it is necessary to work with three types of state variables:

The physical state R_t - This is a snapshot of the status of the physical resources we are managing and their attributes. This might include the amount of water in a reservoir,

the price of a stock or the location of a sensor on a network. It could also refer to the location and speed of a robot.

The information state I_t - This encompasses any other information we need to make a decision, compute the transition or compute the objective function. We can think of I_t as information about quantities and parameters that we know perfectly, but which do not seem to belong in the physical state R_t which typically captures resources we are managing.

The belief (or knowledge) state B_t - The belief state is information specifying a probability distribution describing an unknown quantity or parameter. The type of distribution (e.g. binomial, normal, or exponential) is typically specified in the initial state S_0, although there are exceptions to this. The belief state B_t is information just like R_t and I_t, except that it is information specifying a probability distribution (such as the mean and variance of a normal distribution), or the statistics characterizing a frequentist model (see sections 3.3 and 3.4).

We then pull these together to create our state variable

$$S_t = (R_t, I_t, B_t).$$

Mathematically, the information state I_t should include information about resources R_t, since R_t is, after all, a form of information. The distinction between I_t (such as wind speed, temperature or the stock market), and R_t (how much energy is in the battery, water in a reservoir or money invested in the stock market) is not important. We separate the variables simply because there are so many problems that involve managing physical or financial resources, and it is often the case that decisions impact only the physical resources. At the same time, B_t includes probabilistic information about parameters that we do not know perfectly. Knowing a parameter perfectly, as is the case with R_t and I_t, is just a special case of a probability distribution.

A proper representation of the relationship between B_t, I_t and R_t is illustrated in figure 9.3. However, we find it more useful to make a distinction (even if it is subjective) of what constitutes a variable that describes part of the physical state R_t, and then let I_t be all remaining variables that describe quantities that are known perfectly. Then, we let B_t consist entirely of probability distributions that describe parameters that we do not know perfectly.

State variables take on different flavors depending on the mixture of physical, informational and knowledge states, as well as the relationship between the state of the system now, and the states in the past.

- Physical state - There are three important variations that involve a physical state:
  - Pure physical state - There are many problems which involve only a physical state which is typically some sort of resource being managed. There are problems where R_t is a vector, a low-dimensional vector (as in $R_t = (R_{ti})_{i \in \mathcal{I}}$ where i might be a blood type, or a type of piece of equipment), or a high-dimensional vector (as in $R_t = (R_{ta})_{a \in \mathcal{A}}$ where a is a multidimenisonal attribute vector).
  - Physical state with information - We may be managing the water in a reservoir (captured by R_t) given temperature and wind speed (which affects evaporation) captured by I_t.

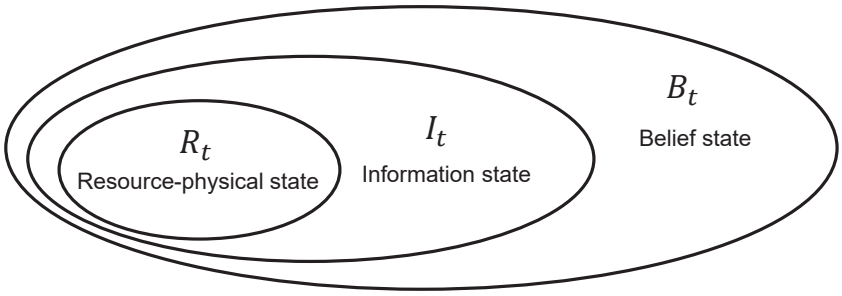


Figure 9.3 Illustration of the growing sets of state variables, where information state includes physical state variables, while the belief state includes everything.

  - Physical state, information state, and belief state - We need the cash on hand in a mutual fund, R_t, information I_t about interest rates, and a probability model B_t describing, say, our belief about whether the stock market is going up or down.

- Information state - In most applications information evolves exogenously, although there are exceptions. The evolution of information comes in several flavors:
  - Memoryless - The information I_{t+1} does not depend on I_t. For example, we may feel that the characteristics of a patient arriving at time $t+1$ to a doctor's office is independent of the patient arriving at time t. We may also believe that rainfall in month $t+1$ is independent of the rainfall in month t.
  - First-order Markov - Here we assume that I_{t+1} depends on I_t. For example, we may feel that the spot market price of oil, the wind speed, or temperature and humidity at $t+1$ depend on the value at time t. We might also insist that a decision x_{t+1} not deviate more than a certain amount from the decision x_t at time t.
  - Higher-order Markov - We may feel that the price of a stock p_{t+1} depends on p_t, p_{t-1}, and p_{t-2}. However, we can create a variable $\bar{p}_t = (p_t, p_{t-1}, p_{t-2})$ and convert such a system to a first-order Markov system, so we really only have to deal with memoryless and first-order Markov systems.
  - Full history dependent - This arises when the evolution of the information I_{t+1} depends on the full history, as might happen when modeling the progress of currency prices or the progression of a disease. This type of model is typically used when we are not comfortable with a compact state variable (and there are methods designed to handle these problems - see 19.9).

- Belief state - Belief states capture beliefs we have about uncertain quantities or parameters that are (typically) evolving over time, often as a direct or indirect results of a decision. Uncertainty in the belief state can arise in three ways:
  - Uncertainty about a static parameter - For example, we may not know the impact of price on demand, or the sales of a laptop with specific features. The nature of the unknown parameter depends on the type of belief model: the

features of the laptop correspond to a lookup table, while the demand-price tradeoff represents the parameter of a parametric model. These problems are broadly known under the umbrella of optimal learning, but are often associated with the literature on multiarmed bandit problems.

- Uncertainty about a dynamic (uncontrollable) parameter - The sales of a laptop with a specific set of features may change over time. This may occur because of unobservable variables. For example, the demand elasticity of a product (such as housing) may depend on other market characteristics (such as the growth of industry in the area).
- Uncertainty about a dynamic, controllable parameter - Imagine that we control the inventory of a product that we cannot observe perfectly. We may control purchases that replenish inventory which is then used to complete sales, but our ability to track sales is imperfect, giving us an imprecise estimate of the inventory. These problems are typically referred to as partially observable Markov decision processes (POMDPs).

There has been a tendency in the literature to treat the belief state as if it were somehow different than "the" state variable. It is not. The state variable is all the information that describes the system at time t, whether that information is the amount of inventory, the location of a vehicle, the current weather or interest rates, or the parameters of a distribution describing some unknown quantity. It the decision maker only has a belief about an uncertain parameter, then for that decision problem, the belief is very much a part of the state variable.

We believe we resolve this unique point of confusion in chapter 20 (Multiagent modeling and learning) by offering a two-agent model (the environment and the controlling agent), which means there are two state variables: one for the environment, and one for the controlling agent. When making a decision, the controlling agent only has access to what is in their state variable, and if this is a belief about an uncertain quantity, then we work with this, just as we did in chapter 7 (think of the interval estimation policy).

We can use S_t to be the state of a single resource (if this is all we are managing), or let $S_t = R_t$ be the state of all the resources we are managing. There are many problems where the state of the system consists only of R_t. We suggest using S_t as a generic state variable when it is not important to be specific, but it must be used when we may wish to include other forms of information. For example, we might be managing resources (consumer products, equipment, people) to serve customer demands $\hat{D}_t$ that become known at time t. If R_t describes the state of the resources we are managing, our state variable would consist of $S_t = (R_t, \hat{D}_t)$, where $\hat{D}_t$ represents additional information we need to solve the problem.

9.4.3 Initial state S_0 vs. subsequent states $S_t,\ t > 0$

It is important to distinguish between the initial state S_0 and subsequent states $S_t,\ t > 0$, as we explain:

The initial state S_0 - The initial state plays a special role in the modeling of a sequential decision problem. It stores any data that is an input to the system, which may include:

- Any deterministic parameters - This might include the deterministic data describing a graph (for example), or any problem parameters that never change.
- Initial values of parameters that evolve over time - For example, this could be the initial inventory, the starting location of a robot, or the initial speed of wind at a wind farm.
- The distribution of belief about uncertain parameters - This is known as the *prior* distribution of belief about anything that is not known perfectly. We emphasize that this prior can be a Bayesian prior, or the initial statistics of a frequentist model.

The subsequent states S_t, $t > 0$ - By convention, the dynamic state S_t (for $t > 0$) only contains the information that changes over time. Thus, if we were solving a shortest path problem over a deterministic graph, S_t would tell us the node which we currently occupy, but would not include, for example, the deterministic data describing the graph which is not changing (by assumption) as we move over the graph. Similarly, it would not include any deterministic parameters such as the maximum speed of our vehicle.

As our system evolves, we drop any deterministic parameters that do not change. These become *latent* (or hidden) variables, since our problem depends on them, but we drop them from S_t for $t > 0$. However, it is important to recognize that these values may change each time we solve an instance of the problem. Examples of these random starting states include:

■ **EXAMPLE 9.1**

We wish to optimize the management of a fleet of trucks. We fix the number of trucks in our fleet, but this is a parameter that we specify, and we may change the fleet size from one instance of the problem to another.

■ **EXAMPLE 9.2**

We wish to optimize the amount of energy to store in a battery given a forecast of clouds over a 24-hour planning horizon. Let $f_{0t'}$ is the forecast of energy at time t' which is given to us at time 0, the vector of forecasts $f_0 = (f_{0t'})_{t'=0}^{24}$ (which does not evolve over time) is part of the initial state. However, each time we optimize our problem, we are given a new forecast.

■ **EXAMPLE 9.3**

We are designing an optimal policy for finding the best medication for type II diabetes, but the policy depends on the attributes of the patient (age, weight, gender, ethnicity, and medical history), which do not change over the course of the treatment.

9.4.4 Lagged state variables*

There are a number of settings where our state variable is actually telling us information about the future. The simplest example arises in resource allocation problems, where resources (trucks/trains/planes enroute to a destination, inbound inventory, people undergoing training) are known now, but will not be available to be used until some point in the

future. We would capture this using

$$\begin{aligned} R_{tt'} &= \text{The resources on hand at time } t \text{ that cannot be used until time } t', \\ R_t &= (R_{tt'})_{t' \geq t}. \end{aligned}$$

Another example would be customer orders being made at time t to be served in the future. For example, we might have

$$\begin{aligned} D_{tt'} &= \text{The number of reservations to fly on an airplane at time } t' \text{ that we know about at time } t, \\ D_t &= (D_{tt'})_{t' \geq t}. \end{aligned}$$

Both R_t and D_t would be considered part of our state S_t.

9.4.5 The post-decision state variable*

Our standard strategy is to model the state variable S_t as all the information we need to make a decision (as well as computing costs, constraints and the transition function). This allows us to write the sequence of state, decision, information as

$$(S_0, x_0, W_1, S_1, x_1, W_2, S_2, x_2, \ldots, x_{t-1}, W_t, S_t). \tag{9.13}$$

Since the state S_t is what we know just before we make a decision, we might also refer to it as the *pre-decision state.* There are settings where we will find it useful to model the state immediately after we make a decision. We model this as S_t^x to indicate that it is still being observed at time t, but immediately after we make the decision x (hence the superscript). We refer to S_t^x as the *post-decision state*. Our information sequence (9.13) becomes

$$(S_0, x_0, S_0^x, W_1, S_1, x_1, S_1^x, W_2, S_2, x_2, S_2^x, \ldots, x_{t-1}, S_{t-1}^x, W_t, S_t). \tag{9.14}$$

Since there is no new exogenous information between making the decision x_t and the observation of the post-decision state S_t^x, the post-decision state is a deterministic function of the pre-decision state S_t and x_t.

The examples below provide some illustrations of pre- and post-decision states.

EXAMPLE 9.4

A traveler is driving through a network, where the travel time on each link of the network is random. As she arrives at node i, she is allowed to see the travel times on each of the links out of node i, which we represent by $\hat{\tau}_i = (\hat{\tau}_{ij})_j$. As she arrives at node i, her pre-decision state is $S_t = (i, \hat{\tau}_i)$. Assume she decides to move from i to k. Her post-decision state is $S_t^x = (k)$. Note that she is still at node i; the post-decision state captures the fact that she will next be at node k, and we no longer have to include the travel times on the links out of node i.

EXAMPLE 9.5

The nomadic trucker revisited. Let $R_{ta} = 1$ if the trucker has attribute vector a at time t and 0 otherwise. Now let D_{tb} be the number of customer demands (loads of

freight) of type b available to be moved at time t. The pre-decision state variable for the trucker is $S_t = (R_t, D_t)$, which tells us the state of the trucker and the demands available to be moved. Assume that once the trucker makes a decision, all the unserved demands in D_t are lost, and new demands become available at time $t+1$. The post-decision state variable is given by $S_t^x = R_t^x$ where $R_{ta}^x = 1$ if the trucker has attribute vector r after a decision has been made.

EXAMPLE 9.6

Imagine playing backgammon where R_{ti} is the number of your pieces on the i^{th} "point" on the backgammon board (there are 24 points on a board). The transition from S_t to S_{t+1} depends on the player's decision x_t, the play of the opposing player, and the next roll of the dice. The post-decision state variable is simply the state of the board after a player moves but before his opponent has moved.

The post-decision state can be particularly valuable in the context of dynamic programming, which we are going to address in depth in chapters 16 and 17.

There are three ways of finding a post-decision state variable:

Decomposing decisions and information There are many problems where we can create functions $S^{M,x}(\cdot)$ and $S^{M,W}(\cdot)$ from which we can compute

$$S_t^x = S^{M,x}(S_t, x_t), \tag{9.15}$$

$$S_{t+1} = S^{M,W}(S_t^x, W_{t+1}). \tag{9.16}$$

The structure of these functions is highly problem-dependent. However, there are sometimes significant computational benefits, primarily when we face the problem of making a decision when we are in state S_t, and would like to know the value of the state the decision takes us to. The post-decision state is a deterministic function of the pre-decision state S_t and the decision x_t, which can be computationally very convenient (see chapters 15 and 16).

State-decision pairs A very generic way of representing a post-decision state is to simply write

$$S_t^x = (S_t, x_t).$$

Figure 9.4 provides a nice illustration using our tic-tac-toe example. Figure 9.4a shows a tic-tac-toe board just before player O makes his move. Figure 9.4b shows the augmented state-decision pair, where the decision (O decides to place his move in the upper right hand corner) is distinct from the state. Finally, figure 9.4c shows the post-decision state. For this example, the pre- and post-decision state spaces are the same, while the augmented state-decision pair is nine times larger.

The augmented state (S_t, x_t) is closely related to the post-decision state S_t^x (not surprising, since we can compute S_t^x deterministically from S_t and x_t). But computationally, the difference is significant. If $\mathcal{S}$ is the set of possible values of S_t, and $\mathcal{X}$ is the set of possible values of x_t, then our augmented state space has size $|\mathcal{S}| \times |\mathcal{X}|$, which is obviously much larger (especially if x is a vector!).

The augmented state variable is used in a popular class of algorithms known as Q-learning (which we first introduced in chapter 2), where the challenge is to statistically

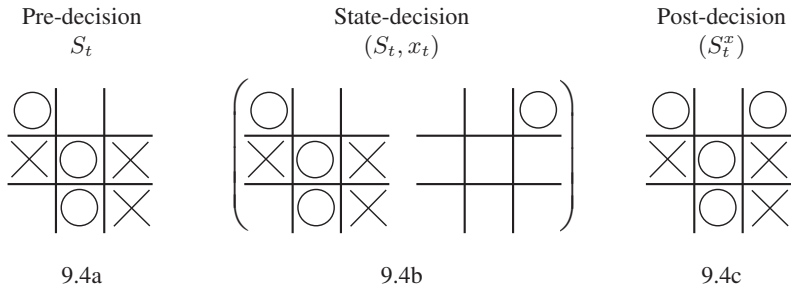


Figure 9.4 Pre-decision state, augmented state-decision, and post-decision state for tic-tac-toe.

estimate Q-factors which give the value of being in state S_t *and* taking decision x_t. The Q-factors are written $Q(S_t, x_t)$, in contrast with value functions $V_t(S_t)$ which provide the value of being in a state. This allows us to directly find the best decision by solving $\min_x Q(S_t, x_t)$. This is the essence of Q-learning, but the price of this algorithmic step is that we have to estimate $Q(S_t, x_t)$ for each S_t and x_t. It is not possible to determine x_t by optimizing a function of S_t^x alone, since we generally cannot determine which decision x_t brought us to S_t^x.

The post-decision as a point estimate Assume that we have a problem where we can compute a point estimate of future information. Let $\overline{W}_{t,t+1}$ be a point estimate, computed at time t, of the outcome of W_{t+1}. If W_{t+1} is a numerical quantity, we might use $\overline{W}_{t,t+1} = \mathbb{E}(W_{t+1}|S_t)$ or $\overline{W}_{t,t+1} = 0$.

If we can create a reasonable estimate $\overline{W}_{t,t+1}$, we can compute post- and pre-decision state variables using

$$\begin{aligned} S_t^x &= S^M(S_t, x_t, \overline{W}_{t,t+1}), \\ S_{t+1} &= S^M(S_t, x_t, W_{t+1}). \end{aligned}$$

Measured this way, we can think of S_t^x as a point estimate of S_{t+1}, but this does not mean that S_t^x is necessarily an approximation of the expected value of S_{t+1}.

9.4.6 A shortest path illustration

We are going to use a simple shortest path problem to illustrate the process of defining a state variable. We start with a deterministic graph shown in figure 9.5, where we are interested in finding the best path from node 1 to node 11. Let t be the number of links we have traversed, and let N_t be the node number were we are located after $t = 2$ transitions. What state are we in?

Most people answer this with

$$S_t = N_t = 6.$$

This answer hints at two conventions that we use when defining a state variable. First, we exclude any information that is not changing, which in this case is any information about our deterministic graph. It also excludes the prior nodes in our path (1 and 3) since these are not needed for any future decisions.

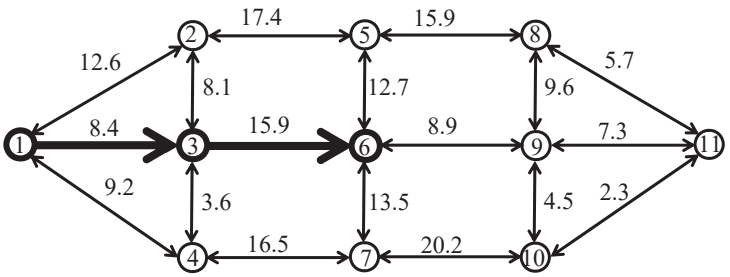


Figure 9.5 A deterministic network for a traveler moving from node 1 to node 11 with known arc costs.

Now assume that the travel times are random, but where we know the probability distribution of travel times over each link (and these distributions are not changing over time). This graph is depicted in Figure 9.6. We are going to assume, however, that when a traveler arrives at node i, she is able to see the actual cost $\hat{c}_{ij}$ for the link (i, j) out of node i (if this is the link that is chosen now). Now, what is our state variable?

Obviously, we still need to know our current node $N_t = 6$. However, the revealed link costs also matter. If the cost of moving from node 6 to node 9 changes from 9.7 to 2.3 or 18.4, our decision may change. This means that these costs are very much a part of our state of information. Thus, we would write our state as

$$S_t = (\underbrace{N_t}_{R_t}, \underbrace{(\hat{c}_{N_t,\cdot})}_{I_t}) = (\underbrace{6}_{R_t}, \underbrace{(10.2, 9.7, 11.8)}_{I_t}),$$

where $(\hat{c}_{N_t,\cdot})$ represents the costs on all the links out of node N_t. Thus, we see an illustration of both a physical state $R_t = N_t$, and information $I_t = (10.2, 9.7, 11.8)$.

For our last example, we introduce the problem of left-hand turn penalties. If our turn from node 6 to node 5 is a left hand turn, we are going to add a penalty of .7 minutes. Now what is our state variable?

The left-hand turn penalty requires that we know if the move from 6 to 5 is a left hand turn. This calculation requires knowing where we are coming from. Thus, we now need to

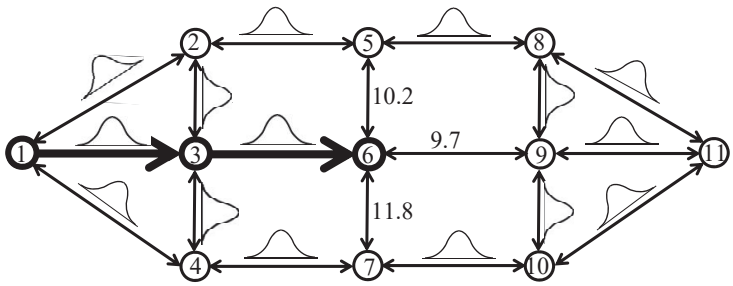


Figure 9.6 A stochastic network, where arc costs are revealed as the traveler arrives to a node.

include our previous node, N_{t-1} in our state variable, giving us

$$S_t = (\underbrace{N_t}_{R_t}, \underbrace{(\hat{c}_{N_t,\cdot}), N_{t-1}}_{I_t}) = (\underbrace{6}_{R_t}, \underbrace{(10.2, 9.7, 11.8), 3}_{I_t}).$$

Now, N_t is our physical state, but N_{t-1} is a piece of information required to compute the cost function.

9.4.7 Belief states*

There are many applications where we are not able to observe (or measure) the state of the system precisely. Instead, we will maintain a probabilistic belief about the unknown parameter or quantity. Some examples include:

EXAMPLE 9.7

A patient may have cancer in the colon which might be indicated by the presence of polyps (small growths in the colon). The number of polyps is not directly observable. There are different methods for testing for the presence of polyps that allow us to infer how many there may be, but these are imperfect.

EXAMPLE 9.8

The military has to make decisions about sending out aircraft to remove important military targets that may have been damaged in previous raids. These decisions typically have to be made without knowing the precise state of the targets.

EXAMPLE 9.9

Policy makers have to decide how much to reduce CO2 emissions, and would like to plan a policy over 200 years that strikes a balance between costs and the rise in global temperatures. Scientists cannot measure temperatures perfectly (in large part because of natural variations), and the impact of CO2 on temperature is unknown and not directly observable.

For each of these examples, we have a system with quantities or parameters that cannot be observed, along with variables that can be observed. When this happens, we handle these values using our belief state B_t.

There is an extensive literature on what are known as "partially observable Markov decision processes" (or "POMDPs") which are sequential decision problems with quantities or parameters that are not known perfectly. The POMDP literature is both mathematically sophisticated as well as computationally challenging. In other words, once you figure out the math, you do not end up with tools that can solve real problems.

Our belief is that the POMDP literature is not modeling these problems correctly. We believe they should be modeled as multiagent systems where there is an "environment agent" and a "controlling agent" that cannot observe the environment perfectly. When models of each agent are formulated with our modeling framework, problems with belief

states become much more practical. We defer this discussion to our chapter on multiagent problems in chapter 20.

9.4.8 Latent variables*

One of the more subtle dimensions of any dynamic model is the presence of information that is not explicitly captured in the state variable S_t. Remember that we do not model in the dynamic state S_t any information in S_0 that does not change over time. We may have many static parameters in S_0. While these are used in the model, they are not in S_t. An optimal policy will depend on this information, but the dependence is not explicit.

Below are some examples of observable latent variables:

■ **EXAMPLE 9.10**

We are solving a shortest path from from origin node r to destination node s, and let $i, j \in \mathcal{I}$ be intermediate nodes. If we are currently at node i, we would model our "state" as being at node i. We use a shortest path algorithm to find the best path from each $i \in \mathcal{I}$, which would give us the "value" $V(i)$ (really the cost) of being at each node i and following the optimal path to node s. The destination s is actually a latent variable, since it is not captured in our state. If we included s in our state variable, we would have to compute the optimal value $V(i, s)$ for every combination of i and s, which is much harder.

■ **EXAMPLE 9.11**

A company is optimizing inventories at different distribution centers. Each DC optimizes its own inventory given the demands that it sees. However, since orders can be satisfied from any DC (not always the closest), the optimal inventory at any single DC depends on the inventories at the other DCs, which become latent variables in the planning of each DC.

■ **EXAMPLE 9.12**

A drug company is determining the optimal dosage given the weight of a patient. The right dosage also depends on age, as well as medical conditions such as diabetes. A dosage table that depends on weight, age and blood sugar would be too complicated, but a patient's physician would need to consider these variables. This means that the physician needs to optimize the dosage for each patient, starting with the dosage table from the drug company which ignores variables other than weight.

When deciding whether to model a variable explicitly (which means modeling how it evolves over time) or as a latent variable (which means holding it constant) introduces an important tradeoff: including a dynamically varying parameter in the state variable produces a more complex, higher dimensional state variable, but one which does not have to be reoptimized when the parameter changes. By contrast, treating a parameter as a latent variable simplifies the model, but requires that the model be reoptimized when the parameter changes.

9.4.9 Rolling forecasts*

An important dimension of any dynamic model is the probability distribution of random activities that will happen in the future. For example, imagine that we are planning the commitment of energy resources given a forecast $f^W_{tt'}$ made at time t of the energy $W_{t'}$ that will be generated from wind at some time t' in the future. We might now assume that the wind in the future is given by

$$W_{t'} = f^W_{tt'} + \varepsilon_{tt'},$$

where $\varepsilon_{tt'} \sim N(0, (t'-t)\sigma^2)$.

For this simple model, where the variance of the error depends only on how far into the future we are projecting, the forecast determines the probability distribution of the energy from wind. In the vast majority of models, we treat the forecast $(f^W_{tt'})^{t+H}_{t'=t}$ as fixed. This means that our forecast is a form of latent variable. In practice, the forecast evolves over time as new information arrives. We might model this evolution using

$$f^W_{t+1,t'} = f^W_{tt'} + \hat{f}^W_{t+1,t'}, \tag{9.17}$$

where $\hat{f}^W_{t+1,t'} \sim N(0, \sigma^2_W)$ represents the exogenous change in the forecast for time t'. Equation (9.17) is known as a *martingale model of forecast evolution*, or MMFE, in the inventory literature, which means that the expected value of the forecast in the future, $f^W_{t'',t'}$, is equal to the forecast $f^W_{t,t'}$ now. [A "martingale" is a stochastic process that might evolve up or down from one time period to the next, but on average stays the same.] This means that if we are given $f^W_{tt'}, t' = t, \ldots, T$, then this vector of forecasts is a part of the state variable, since all of this information is needed to model the evolution of W_t.

For many problems, however, forecasts are not modeled as a dynamically evolving stochastic process; instead, they are viewed as static, which means that they are not part of the state variable. In this case they would be *latent variables*, and are not explicitly being modeled. As we see in chapter 19 on direct lookahead models, holding forecasts constant is a common approximation in direct lookahead models. This is what navigation systems are doing when planning a path over the network. At a point in time, the system fixes the estimate of the travel time over each link and plans a path. A few minutes later, the times are updated and the path is recalculated, but the logic for finding the path does not explicitly model the possible changes in the estimates of link times.

Classical dynamic programming models seem to almost universally ignore the role of forecasts in the modeling of a dynamic optimization problem, which means that they are being treated as latent variables. This in turn means that the problem has to be re-optimized from scratch when the forecasts are updated. By contrast, forecasts are easily handled in lookahead policies, as we see later. We describe methods for handling forecasts in chapter 19; for now, we just want to show how the presence of rolling forecasts can affect the state variable.

9.4.10 Flat vs. factored state representations*

It is very common in the dynamic programming literature to define a discrete set of states $\mathcal{S} = (1, 2, \ldots, |\mathcal{S}|)$, where $s \in \mathcal{S}$ indexes a particular state. For example, consider an inventory problem where S_t is the number of items we have in inventory (where S_t is a scalar). Here, our state space $\mathcal{S}$ is the set of integers, and $s \in \mathcal{S}$ tells us how many products are in inventory.

Now assume that we are managing a set of K product types. The state of our system might be given by $S_t = (S_{t1}, S_{t2}, \ldots, S_{tk}, \ldots)$ where S_{tk} is the number of items of type k in inventory at time t. Assume that $S_{tk} \leq M$. Our state space $\mathcal{S}$ would consist of all possible values of S_t, which could be as large as K^M. A state $s \in \mathcal{S}$ corresponds to a particular vector of quantities $(S_{tk})_{k=1}^K$.

Modeling each state with a single scalar index is known as a flat or unstructured representation. Such a representation is simple and elegant, and produces mathematically compact models that have been popular in communities like operations research and computer science. We first saw this used in section 2.1.3, and we will return to this in chapter 14 in much more depth. However, the use of a single index completely disguises the structure of the state variable, and often produces intractably large state spaces.

In the design of algorithms, it is often essential that we exploit the structure of a state variable. For this reason, we generally find it necessary to use what is known as a *factored* representation, where each factor represents a *feature* of the state variable. For example, in our inventory example we have K factors (or features). It is possible to build approximations that exploit the structure that each dimension of the state variable is a particular quantity.

In section 8.3.1, we solved a problem of managing resources (people, equipment) which were described by an attribute vector $a \in \mathcal{A}$, where we assumed that the attribute space $\mathcal{A}$ was discrete. This is an example of a flat representation. Each element a_i of an attribute vector represents a particular feature of the entity. This representation allowed us to model the resource vector as $R_t = (R_{ta})_{a \in \mathcal{A}}$, where R_t is now a vector with element $a \in \mathcal{A}$.

9.4.11 A programmer's perspective of state variables

State variables are easily one of the least understood concepts in dynamic modeling, as evidenced by the large number of books on dynamic programs that do not even define a state variable (at least not properly). For a different perspective, imagine that you are programming a simulator of a dynamic system where decisions are made over time. You are going to create a set of variables to model your system (for many problems, this can be a lot of variables). We can divide the variables into four broad categories:

Category 1 - These are all the variables that are set initially (either hard coded into the program, or read in from an external data source). We can divide these into several subcategories:

- **1a)** - Fixed parameters (such as the boiling point of water, or the maximum speed of a vehicle) that never change.
- **1b)** - Initial values of variables that evolve over time (whether due to decisions and/or exogenous inputs).
- **1c)** - Initial beliefs about parameters and quantities that are not known perfectly. These beliefs may or may not evolve over the course of the simulation.

Category 2 - Variables that change over the course of the simulation, either due to decisions or exogenous inputs (and we exclude decisions and exogenous information which we put in categories 3 and 4 below). These can include:

- **2a)** - Variables that describe quantities and parameters that are known perfectly.
- **2b)** - Variables that describe probability distributions that evolve over time. These variables might describe the parameters of parametric distributions, probabilities, or sufficient statistics.

Category 3 - Variables that represent decisions that are determined by some policy.

Category 4 - Information that enters our system exogenously. This information may be used to make a decision and discarded, or may become included in a category 2 variable.

All the variables that fall in Category 1 are what we put in our initial state variable S_0. All the variables that fall in Category 2 are what we put in our initial state variable S_t for $t > 0$. Category 3 refers to variables that we control, known as control variables, actions or, in this book, decisions that we call x_t (see section 9.5 below). Finally, Category 4 refers to new information that arrives from outside our system, which we have modeled as W_{t+1}. We may use this to make a decision and then discard it. However, it may be blended into one of the variables in Category 2.

From this discussion, we see that "state variables" are what a programmer would call a "variable," although we exclude decision variables and exogenous information variables, unless these are retained for future use (in which case they become included in Category 2). Also, a programmer may retain a lot of information in "variables" for reporting purposes, whereas we restrict our definition of state variables to information that we actually need to model our system.

9.5 MODELING DECISIONS

There are a number of words in English that can mean "decision," as illustrated in table 9.1. The optimization literature assumes that the types of decisions are known in advance, overlooking what can be one of the most subtle dimensions of modeling. Arguably one of the most challenging dimensions of optimization is recognizing exactly what decisions need to be optimized!

It should not be surprising that even the optimization communities use different words (and notation) to mean decision. The classical literature on Markov decision process talks

General terms	Collecting information
Action	Examine
Acquire/buy/purchase	Experiment
Choice	Listen
Control	Observe
Decision	Probe
Design	Research
Intervention (medical)	Sample
Option	Sense
Move	Test
Response	View
Task	Scan
Trade (finance)	

Table 9.1 Sample of words in English that represent a decision. The second column describes decisions in the context of collecting information, as in choice of experiment to run, or what to listen to or observe.

about choosing an action $a \in \mathcal{A}$ (or $a \in \mathcal{A}_s$, where $\mathcal{A}_s$ is the set of actions available when we are in state s). The optimal control community chooses a control $u \in \mathcal{U}_x$ when the system is in state x. The math programming community wants to choose a decision represented by the vector x. We have also noticed that the bandit community in computer science has also adopted "x" as its notation for a decision which is typically discrete. In this book, we use "x" as our default notation, although we occasionally slip back to using action a (in particular, see chapter 14) when we are using methods where the action must be discrete.

When we model decisions in a sequential decision problem, we recommend introducing the following elements:

- The types of decisions, and notation for whether a decision is made (and if appropriate, how much).
- Constraints on decisions made at time t.
- The notation for a policy (or method) for making decisions, but without specifying the policy.

9.5.1 Types of decisions

Decisions come in many forms. We illustrate this using our notation x which tends to be the notation of choice for more complex problems. Examples of different types of decisions are

- Binary, where x can be 0 or 1.
- Discrete set, where $x \in \{x_1, \ldots, x_M\}$.
- Continuous scalar, where $x \in [a, b]$.
- Continuous vector, where $x \in \Re^n$.
- Integer vector, where $x \in \mathbb{Z}^n$.
- Subset selection, where x is a vector of 0's and 1's, indicating which members are in the set.
- Multidimensional categorical, where $x_a = 1$ if we make a choice described by an attribute $a = (a_1, \ldots, a_K)$. For example, a could be the attributes of a drug or patient, or the features of a movie.

There are many applications where a decision is either continuous or vector-valued. For example, in chapter 8 we describe applications where a decision at time t involves the assignment of resources to tasks. Let $x = (x_d)_{d \in \mathcal{D}}$ be the vector of decisions, where $d \in \mathcal{D}$ is a *type* of decision, such as assigning resource i to task j, or purchasing a particular type of equipment. It is not hard to create problems with hundreds, thousands and even tens of thousands of *dimensions*. These high-dimensional decision vectors arise frequently in the types of resource allocation problems addressed in operations research.

This discussion makes it clear that the complexity of the space of decisions (or actions or controls) can vary considerably across applications. There are entire communities dedicated to problems with a specific class of decisions. For example, optimal stopping problems feature binary actions (hold or sell). The entire field of Markov decision processes, as

well as all the problems described in chapter 7 for derivative-free stochastic optimization, assume discrete sets. Derivative-based stochastic optimization, as well as the field of stochastic programming, assumes that x is a vector, usually continuous.

9.5.2 Initial decision x_0 vs. subsequent decisions $x_t,\ t > 0$

Just as we distinguished between the initial state S_0 and subsequent states $S_t,\ t \geq 0$, it is useful to distinguish between the first decision x_0 and ongoing decisions $x_t,\ t > 0$:

Initial decision x_0 - The first decision x_0 is a mixture of initial design decisions that are only made once, and the first instances of ongoing control decisions. Examples of initial design decisions include:

- Location and capacity of fixed facilities.
- The configuration of a manufacturing system or network.
- The design of a robot or other machines.
- The people who are hired to staff the system.
- The initial location and quantities of resources (robots, trucks, nurses) that will be managed over the course of a simulation.
- The parameters that govern the behavior of policies.

All of these are parameters that can be viewed as design variables to be optimized. Particularly important is recognizing that the design of the policy is no different than any of the other decisions that affect the design of a system.

Subsequent decisions $x_t,\ t > 0$ - The decisions x_t represents the decisions that are controlling the system that are made on an ongoing basis. The array of controlling decisions is much too long to list, but we can characterize them in broad categories:

- Decisions which manage physical resources: people, robots, machinery, inventories (of any product), water, energy.
- Decisions which manage financial resources: investments, contracts.
- Decisions that affect the performance of a process: prices, speeds, temperatures.
- Information collection decisions from computer simulations, laboratory experiments, field experiments.
- Decisions to communicate or share information: ads, marketing, promotions.

Feel free to jump back to table 1.1 in chapter 1 for a hint at the diversity of control decisions.

9.5.3 Strategic, tactical and execution decisions

It is important to recognize that there are often lags between when a decision is made (which determines its information content) and when it is implemented (which is the point at which it impacts our system). To handle lagged decision processes we define

$x_{tt'}$ = a decision made at time t to be implemented at time $t' \geq t$.

We now describe three classes of decisions based on the lag:

- Strategic planning - x_0 refers to all decisions made at time $t = 0$. These are our design decisions discussed above.
- Tactical planning - $x_{tt'}$ where $t' > t$ - These are decisions now that impact the future, which means we have to model exogenous information $W_{t+1}, \ldots, W_{t'}$, as well as the decisions $x_{t+1}, \ldots, x_{t'}$ that we make between t and t'.
- Execution - x_{tt} - These are decisions that we implement at time t.

Each of these decisions require simulating other decisions. For example

- Strategic planning - We will need to simulate decisions $x_1, x_2, \ldots, x_T$ in order to evaluate the performance of the design decisions x_0.
- Tactical planning - Here we are making a decision $x_{tt'}$ at time t to implement at time t', which means we need to simulate the decisions $x_t, x_{t+1}, \ldots, x_{t'-1}$ to anticipate the state that we will be in at time t' when we make a decision at time t.
- Execution - To help us make the decision x_{tt} that we are going to implement now (at time t), we will often need to simulate the downstream impact of this decision, which means simulating the decisions $x_{t+1}, x_{t+2}, \ldots, x_T$.

9.5.4 Constraints

When we make decisions at time t, we often have to specify constraints on the decisions. The simplest type of "constraint" is to specify a set of possible (discrete) decisions $\mathcal{D}_s$ given that we are in state s. Often the set of possible types of decisions $\mathcal{D}$ is static, but if it depends on the state (which can vary over time), we would write

$$\begin{aligned} \mathcal{D}_t &= \text{the set of types of decisions given that we are in state } S_t \text{ at time } t. \\ &\quad \text{The dependence on the state } S_t \text{ is implicit through our indexing the} \\ &\quad \text{set by time,} \\ x_{td} &= \text{the number of times we execute decision } d \in \mathcal{D}_t \text{ at time } t. \end{aligned}$$

An example can be assigning drivers to loads at time t, where $\mathcal{D}_t$ is the set of loads available at time t.

If we have a vector of decisions x_{td} for $d \in \mathcal{D}$, we may easily have constraints on the vector x_t. For example, x_{td} might be the amount we invest in stock d, but we have to limit our investments to the amount of cash R_t we have on hand, so we would write:

$$\begin{aligned} \sum_{d \in \mathcal{D}} x_{td} &\leq R_t, \\ x_{td} &\geq 0. \end{aligned}$$

We can write constraints like this in the general format

$$\begin{aligned} A_t x_t &= R_t, \\ x_t &\geq 0. \end{aligned}$$

Even more general is to write

$$x_t \in \mathcal{X}_t,$$

where $\mathcal{X}_t$ may be a discrete set such as $\{x_1, \ldots, x_M\}$, or the solution to our system of linear equations. When we index a set (or variable) by t as in $\mathcal{X}_t$, this means it depends on information in the state S_t. We do not write it as $\mathcal{X}(S_t)$ just to keep the notation compact.

9.5.5 Introducing policies

The challenge of any optimization problem (including stochastic optimization) is making decisions. In a sequential (stochastic) decision problem, the decision x_t depends on the information available at time t, which is captured by S_t. This means we need a decision x_t for each S_t, which means we need a function $x_t(S_t)$. This function is known as a *policy*, often designated by π. While many authors use $\pi(S_t)$ to represent the policy, we use π to carry the information that describes the function, and designate the function as $X^\pi(S_t)$. If we are using action a_t, we would designate our policy as $A^\pi(S_t)$, or $U^\pi(S_t)$ if we are finding control u_t. Policies may be stationary (as we have written them), or time-dependent, in which case we would write $X_t^\pi(S_t)$.

We introduce the notation for the policy, such as $X^\pi(S_t)$ when we introduce decisions in our model, but we do not make any effort at choosing the policy. This is at the heart of our philosophy:

Model first, then solve.

The choice of policy depends not only on the structure of the problem, but it may even depend on the nature of the data for a particular problem. In chapter 11, we are going to describe an energy storage problem (which we model below) where we show that each of four classes of policies (plus a fifth hybrid) can work best depending on the specific characteristics of a dataset.

Starting in chapter 11, we are going to spend the rest of the book identifying different classes of policies that are suited to problems with different characteristics. Note that it is not an accident that we address the design of policies after we discuss modeling uncertainty in chapter 10, which we "model first, then solve."

9.6 THE EXOGENOUS INFORMATION PROCESS

An important dimension of many of the problems that we address is the arrival of exogenous information, which changes the state of our system. Modeling the flow of exogenous information represents, along with states, the most subtle dimension of modeling a stochastic optimization problem. We sketch the basic notation for modeling exogenous information here, and defer to chapter 10 a more complete discussion of uncertainty.

We begin by noting that this section only addresses the exogenous information that arrives at times $t > 0$. This ignores the initial state S_0 which is an entirely different source of information (which technically is exogenous).

9.6.1 Basic notation for information processes

Consider a problem of tracking the value of an asset. Assume the price evolves according to

$$p_{t+1} = p_t + \hat{p}_{t+1}.$$

Sample path	$t=0$	$t=1$		$t=2$		$t=3$	
ω	p_0	$\hat{p}_1$	p_1	$\hat{p}_2$	p_2	$\hat{p}_3$	p_3
1	29.80	2.44	32.24	1.71	33.95	-1.65	32.30
2	29.80	-1.96	27.84	0.47	28.30	1.88	30.18
3	29.80	-1.05	28.75	-0.77	27.98	1.64	29.61
4	29.80	2.35	32.15	1.43	33.58	-0.71	32.87
5	29.80	0.50	30.30	-0.56	29.74	-0.73	29.01
6	29.80	-1.82	27.98	-0.78	27.20	0.29	27.48
7	29.80	-1.63	28.17	0.00	28.17	-1.99	26.18
8	29.80	-0.47	29.33	-1.02	28.31	-1.44	26.87
9	29.80	-0.24	29.56	2.25	31.81	1.48	33.29
10	29.80	-2.45	27.35	2.06	29.41	-0.62	28.80

Table 9.2 A set of sample realizations of prices (p_t) and the changes in prices ($\hat{p}_t$)

Here, $\hat{p}_{t+1}$ is an exogenous random variable representing the change in the price during time interval $t+1$. At time t, p_t is a number, while (at time t) p_{t+1} is random.

We might assume that $\hat{p}_{t+1}$ comes from some probability distribution such as a normal distribution with mean 0 and variance σ^2. However, rather than work with a random variable described by some probability distribution, we are going to primarily work with sample realizations. Table 9.2 shows 10 sample realizations of a price process that starts with $p_0 = 29.80$ but then evolves according to the sample realization. These samples might come from a mathematical model, or observations from history.

Following standard convention, we index each path by the Greek letter ω (in the example below, ω runs from 1 to 10). At time $t = 0$, p_t and $\hat{p}_t$ is a random variable (for $t \geq 1$), while $p_t(\omega)$ and $\hat{p}_t(\omega)$ are *sample realizations*. We refer to the sequence

$$p_1(\omega),\ p_2(\omega),\ p_3(\omega),\ \ldots, p_T(\omega)$$

as a *sample path* for the prices p_t.

We are going to use "ω" notation throughout this volume, so it is important to understand what it means. As a rule, we will primarily index exogenous random variables such as $\hat{p}_t$ using ω, as in $\hat{p}_t(\omega)$. $\hat{p}_{t'}$ is a random variable if we are sitting at a point in time $t < t'$. $\hat{p}_t(\omega)$ is not a random variable; it is a sample realization. For example, if $\omega = 5$ and $t = 2$, then $\hat{p}_t(\omega) = -0.73$. We are going to create randomness by choosing ω at random. To make this more specific, we need to define

$$\begin{aligned} \Omega &= \text{the set of all possible sample realizations (with } \omega \in \Omega), \\ p(\omega) &= \text{the probability that outcome } \omega \text{ will occur.} \end{aligned}$$

A word of caution is needed here. We will often work with continuous random variables, in which case we have to think of ω as being continuous. In this case, we cannot say $p(\omega)$ is the "probability of outcome ω." However, in all of our work, we will use discrete samples. For this purpose, we can define

$$\hat{\Omega} = \text{a set of discrete sample observations of } \omega \in \Omega.$$

In this case, we can talk about $p(\omega)$ being the probability that we sample ω from within the set $\hat{\Omega}$. Often, we will assume that each element of $\hat{\Omega}$ occurs with equal probability:

$$p(\omega) = \frac{1}{|\hat{\Omega}|}.$$

For more complex problems, we may have an entire family of random variables. In such cases, it is useful to have a generic "information variable" that represents all the information that arrives during time interval t. For this purpose, we define

W_{t+1} = the exogenous information becoming available during time interval $(t, t+1)$.

We might also say that W_{t+1} is the information that first becomes known by time $t+1$, which means it is not known when we make the decision x_t.

W_t may be a single variable, or a collection of variables (travel times, equipment failures, customer demands). We note that while we use the convention of putting hats on variables representing exogenous information ($\hat{D}_t, \hat{p}_t$), we do not use a hat for W_t since this is our only use for this variable, whereas D_t and p_t have other meanings. We always think of information as arriving in continuous time, hence W_t is the information arriving during time interval t, rather than at time t. This eliminates the ambiguity over the information available when we make a decision at time t.

We sometimes need to refer to the *history* of our process, for which we define

$$\begin{aligned}
h_t &= \text{the history of the process, consisting of all the information known through time } t,\\
&= (W_1, W_2, \ldots, W_t),\\
\mathcal{H}_t &= \text{the set of all possible histories through time } t,\\
&= \{h_t(\omega) | \omega \in \Omega\},\\
\Omega_t(h_t) &= \text{the set of all sample paths that correspond to history } h_t,\\
&= \{\omega \in \Omega | h_t(\omega) = h_t\}.
\end{aligned}$$

In some applications, we might refer to h_t as the state of our system, but this is usually a very clumsy representation. However, we will use the history of the process for a specific modeling and algorithmic strategy.

9.6.2 Outcomes and scenarios

Some communities prefer to use the term *scenario* to refer to a sample realization of random information. For most purposes, "outcome," "sample path," and "scenario" can be used interchangeably (although sample path refers to a sequence of outcomes over time). There are many, however, who use the term "scenario" to represent a major event. For example, a company may launch a new product that may receive a market response that can be described as strong, medium or weak. For each of these scenarios, there are still going to be daily fluctuations in sales. We prefer to use "scenario" to refer to the market response (that is, the major event), and "outcome" to capture the variations around the market response.

We recommend denoting the set of scenarios by Ψ, with $\psi \in \Psi$ representing an individual scenario. Then, for a particular scenario ψ, we might have a set of outcomes $\omega \in \Omega$ (or $\Omega(\psi)$) representing various minor events (daily sales volume).

Two examples illustrate this notation:

EXAMPLE 9.13

Planning spare transformers - In the electric power sector, a certain type of transformer was invented in the 1960's. As of this writing, the industry does not really know the failure rate curve for these units (is their lifetime roughly 50 years? 60 years?). Let ψ be the scenario that the failure curve has a particular shape (for example, where failures begin happening at a higher rate around 50 years). For a given scenario ψ (the failure rate curve), ω represents a sample outcome of failures (transformers can fail at any time, although the likelihood they will fail depends on ψ).

EXAMPLE 9.14

Long term contracts for electricity - The price of electricity today depends largely on the price of natural gas. Electricity prices on an hourly basis can be highly volatile, but they average a price that reflects the price of natural gas. This relationship may depend on a) the aggregate production of natural gas (which can depend on government policy) and b) the availability of renewables. We can describe the relative supplies of energy from natural gas and renewables as a scenario ψ, and then model hourly variations as a sample path ω.

9.6.3 Lagged information processes*

There are many settings where the information about a new arrival comes before the new arrival itself, as we saw earlier in state variables. These also happen in exogenous information processes, as illustrated in the following examples:

EXAMPLE 9.15

A customer may make a reservation at time t to be served at time t'.

EXAMPLE 9.16

An orange juice products company may purchase futures for frozen concentrated orange juice at time t that can be exercised at time t'.

EXAMPLE 9.17

A programmer may start working on a piece of coding at time t with the expectation that it will be finished at time t'.

We handle these problems using two time indices, a form that we refer to as the "(t, t')" notation.

Lagged information processes are surprisingly common. Let $\hat{D}_{tt'}$ be the number of customers calling in at time t to book a hotel room at time t'. We can write our set of orders arriving on day t as

$$\begin{aligned}\hat{D}_{tt'} &= \text{the demands that first become known during time interval } t \text{ to be served during time interval } t',\\ \hat{D}_t &= (\hat{D}_{tt'})_{t' \geq t}.\end{aligned}$$

Then, $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_t, \ldots$ is the sequence of orders, where each $\hat{D}_t$ can be orders being called in for different times into the future.

An important class of lagged processes are forecasts. Let

$$\begin{aligned}f^D_{tt'} &= \text{the forecast of the demand } \hat{D}_{t'} \text{ during time interval } t' \text{ made using the information available up through time } t,\\ f^D_t &= (f^D_{tt'})_{t' \geq t}.\end{aligned}$$

An important special case of each of these variables is when $t' = t$. We would describe this version of each of the variables above as follows:

$$\begin{aligned}\hat{D}_{tt} &= \text{The actual demand during time } t,\\ f^D_{tt} &= \text{this is another way of writing } \hat{D}_{tt},\\ R_{tt} &= \text{the resources we know about at time } t \text{ that we can use at time } t.\end{aligned}$$

Note that these variables are now written in terms of the information content. For example, $\hat{D}_{tt'}$ are the demands we know about at time t that will need to be served at time t'. The first time index specifies when the information becomes known.

9.6.4 Models of information processes*

Information processes come in varying degrees of complexity. Needless to say, the structure of the information process plays a major role in the models and algorithms used to solve the problem. Below, we describe information processes in increasing levels of complexity.

State-independent processes

Information might be generated by independent, unintelligent, exogenous processes such as weather, markets, biological processes, chemical reactions, complex simulators, where the information is independent of the state S_t or decision x_t.

EXAMPLE 9.18

A publicly traded index fund has a price process that can be described (in discrete time) as $p_{t+1} = p_t + \sigma\delta$, where δ is normally distributed with mean μ, variance 1, and σ is the standard deviation of the change over the length of the time interval.

EXAMPLE 9.19

Requests for credit card confirmations arrive according to a Poisson process with rate λ. This means that the number of arrivals during a period of length Δt is given by a Poisson distribution with mean $\lambda \Delta t$, which is independent of the history of the system.

The practical challenge we typically face in these applications is that we do not know the parameters of the system. In our price process, the price may be trending upward or downward, as determined by the parameter μ. In our customer arrival process, we need to know the rate λ (which can also be a function of time).

State-independent information processes are attractive because they can be generated and stored in advance, simplifying the process of testing policies. In chapter 19, we will describe an algorithmic strategy based on the use of *scenario trees* which have to be created in advance.

State/action-dependent information processes

There are many problems where the exogenous information W_{t+1} depends on the state S_t and/or the decision x_t. Some illustrations include:

EXAMPLE 9.20

The change in the speed of wind at a wind farm depends on the current speed. If the current speed is low, the change is likely to be an increase. If it is high, the change is likely to be a decrease.

EXAMPLE 9.21

A market with limited information may respond to price changes. If the price drops over the course of a day, the market may interpret the change as a downward movement, increasing sales and putting further downward pressure on the price. The market may also respond to decisions by mutual funds to sell large amounts of stock.

EXAMPLE 9.22

Customers arriving to a bank are served by a group of tellers, where the number of tellers on duty are controlled by a bank manager. The arrival rate of customers depend on the length of the queue (which is the state of our system), which depends on the decisions (made hourly) of how many people to have on duty.

State/action-dependent information processes make it impossible to pre-generate sample outcomes when testing policies. While not a major issue, it complicates comparing policies since we cannot fix the sample outcomes.

State-dependent information processes introduce a subtle notational complication. Following standard convention, the notation ω almost universally refers to a sample path. Thus, $W_t(\omega)$ represents the exogenous information arriving between $t-1$ and t when we are following sample path ω. If we write $S_t(\omega)$, we mean the state we are in at time t

when we are following sample path ω, but now we have to make it clear what policy we are following to get there. For example, we might write $S^{\pi}_{t+1} = S^M(S^{\pi}_t, X^{\pi}_t(S_t), W^{\pi}_{t+1}(\omega))$, where it is clear that we are using policy π to get from S^{π}_t to S^{π}_{t+1}.

Multiagent sytems

The exogenous information may come from the decisions made by another agent. We can make the argument that W_{t+1}, which is really the decisions of another agent, would be a random variable that depends on some observable system state variables (such as the state of a game board), and the decision x_t made by the first agent. However, with enough training, the behavior of each agent tends to become predictable (this is typical of experts playing against each other), which means deterministic (although one strategy in an adversarial game is to introduce noise to keep the opponent from learning your strategies).

We cover the topic of multiagent systems in chapter 20.

More complex information processes

Now consider the problem of modeling currency exchange rates. The change in the exchange rate between one pair of currencies is usually followed quickly by changes in others. If the Japanese yen rises relative to the U.S. dollar, it is likely that the Euro will also rise relative to it, although not necessarily proportionally. As a result, we have a vector of information processes that are correlated.

In addition to correlations between information processes, we can also have correlations over time. An upward push in the exchange rate between two currencies in one day is likely to be followed by similar changes for several days while the market responds to new information. Sometimes the changes reflect long term problems in the economy of a country. Such processes may be modeled using advanced statistical models which capture correlations between processes as well as over time.

An *information model* is a mathematical model of the underlying information process. This falls under the broad umbrella of uncertainty modeling or uncertainty quantification, which we cover in chapter 10. In some cases with complex information models, it is possible to proceed without any model at all. Instead, we can use realizations drawn from history. For example, we may take samples of changes in exchange rates from different periods in history and assume that these are representative of changes that may happen in the future. The value of using samples from history is that they capture all of the properties of the real system. This is an example of planning a system without a model of an information process.

Deterministic models

While listing different types of exogenous information processes, we cannot ignore the possibility that we do not have an exogenous information process, as would be the case with any deterministic system. We note that a large majority of the work in optimal control performed (primarily) in engineering applications is deterministic.

9.6.5 Supervisory processes*

We are sometimes trying to control systems where we have access to a set of decisions from an exogenous source. These may be decisions from history, or they may come from a knowledgeable expert. Either way, this produces a dataset of states $(S^m)_{m=1}^n$ and decisions $(x^m)_{m=1}^n$. In some cases, we can use this information to fit a statistical model which we use to try to predict the decision that would have been made given a state.

The nature of such a statistical model depends very much on the context, as illustrated in the examples:

EXAMPLE 9.23

We can capture data on patient histories and complaints, along with the treatment decisions by physicians. We can use this history to train a neural network to recommend a treatment given the characteristics of a patient.

EXAMPLE 9.24

We can use the history of decisions when playing games (notably video games, but also games such as chess and computer Go), to train a statistical model what decision to make given the state of the game.

We can use supervisory processes to statistically estimate a decision function that forms an initial policy. We can then use this policy in the context of methods to create even better policies using the principles of policy search. The supervisory process helps provide an initial policy that may not be perfect, but at least is reasonable.

9.7 THE TRANSITION FUNCTION

The next step in modeling a dynamic system is the specification of the *transition function* which is a concept that is widely used in the optimal control community. This function describes how the system evolves from one state to another as a result of decisions and information. If you have ever written a simulator of a dynamic system, you have written a transition function, since this is nothing more than the equations that describe how variables evolve over time.

We begin our discussion of system dynamics by introducing some general mathematical notation. While useful, this generic notation does not provide much guidance into how specific problems should be modeled. We then describe how to model the dynamics of some simple problems, followed by a more general model for complex resources.

9.7.1 A general model

The dynamics of our system are represented by a function that describes how the state evolves as new information arrives and decisions are made. The optimal control community will usually write the transition function (using controls notation) as

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

where x_t is their notation for state, u_t is the decision or control, and w_t is the exogenous information which is random at time t (there is a long history behind this). The function $f(\cdot)$ goes by different names such as "plant model" (literally, the model of a physical production plant), "plant equation," "law of motion," "transfer function," "system dynamics," "system model," "state equations," "transition law," as well as "transition function."

When modeling complex problems, the letters f, g and h are widely used for "functions," where f in particular is popular for being used in many ways. To avoid taking this valuable

piece of real estate in the alphabet, we use the notation

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}). \tag{9.18}$$

We use the notation $S^M(\cdot)$ since it hints at "state model" or "state transition model." This style avoids using another letter from the alphabet.

For real-world problems, the transition function often hides tremendous complexity in the modeling of the dynamics of a system. A transition function can easily consist of hundreds or thousands of lines of code. Of course, we started with a simple example in section 9.1 that required only two equations.

This is a very general way of representing the dynamics of a system. Assuming we have a proper state variable S_t that captures all the information we need to model the system from time t onward, the information W_{t+1} arriving during time interval $(t, t+1)$ depends on the state S_t at the end of time interval t (and possibly the decision x_t). In this case, we can store the system dynamics in the form of a one-step transition matrix using

$$P(s'|s,x) \quad = \quad \text{the probability that } S_{t+1} = s' \text{ given } S_t = s \text{ and } X^\pi(S_t) = x.$$

The one-step transition matrix is the foundation of a field known as discrete Markov decision processes, which we cover in chapter 14. There is a simple relationship between the transition function and the one-step transition matrix. Define the indicator function

$$\mathbb{1}_X = \begin{cases} 1 & \text{if } X \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Assuming that the set of outcomes of $W_{t+1} = w \in \Omega^W$ is discrete, the one-step transition matrix can be computed using

$$\begin{aligned} P(s'|s,x) &= \mathbb{E}_{W_{t+1}}\{\mathbb{1}_{\{s'=S^M(S_t=s,x_t=x,W_{t+1})\}}|S_t = s, x_t = x\} \\ &= \sum_{w\in\Omega^W} P(W_{t+1} = w|S_t = s, x_t = x)\mathbb{1}_{\{s'=S^M(S_t=s,x_t=x,w)\}}. \end{aligned} \tag{9.19}$$

We now have two ways of representing the dynamics of our system: the transition function $S^M(S_t, x_t, W_{t+1})$, and the one-step transition matrix $P(s'|s,x)$. The controls community (which is substantial) uses the transition function, while the community that works with Markov decision processes (which was adopted by the reinforcement learning community within computer science) uses the one-step transition matrix $P(s'|s,x)$. Given the derivation above, it seems clear that you need the one-step transition function in order to compute the one-step transition matrix. Yet, the MDP community will often treat the one-step transition matrix as input data.

In this book we exclusively use the one-step transition *function*, since this is trivially computable, even when the state variable S_t is high-dimensional (and even continuous). It is literally the equations you would use to simulate the system. By contrast, the one-step transition matrix is a powerful theoretical device, but it is utterly incomputable for all but the most trivial problems.

9.7.2 Model-free dynamic programming

There are many complex operational problems where we simply do not have a transition function. Some examples include

EXAMPLE 9.25

We are trying to find an effective policy to tax carbon to reduce CO2 emissions. We may try increasing the carbon tax, but the dynamics of climate change are so complex that the best we can do is wait a year and then repeat our measurements.

EXAMPLE 9.26

A ride hailing service encourages drivers to go on duty by raising prices (surge pricing). Since it is impossible to predict how drivers will behave, it is necessary to simply raise the price and observe how many drivers come on duty (or go off duty).

EXAMPLE 9.27

A utility managing a water reservoir can observe the level of the reservoir and control the release of water, but the level is also affected by rainfall, river inflows, and exchanges with ground water, which are unobservable.

These examples illustrate problems where we do not know the dynamics, where the system reflects the unknown utility function of Uber drivers, and unobservable exogenous information. As a result, we either do not know the transition function itself, or there are decisions that we cannot model, or exogenous information we cannot simulate. In all three cases, we cannot compute the transition $S_{t+1} = S^M(S_t, x_t, W_{t+1})$.

In such settings (which are surprisingly common), we assume that given the state S_t, we take an action x_t and then simply observe the next state S_{t+1}. We can put this in the format of our original model by letting W_{t+1} be the new state, and writing our transition function as

$$S_{t+1} = W_{t+1}.$$

However, it is more natural (and compact) to simply assume that our system evolves according to

$$S_0 \rightarrow x_0 \rightarrow S_1 \rightarrow x_1 \rightarrow S_2 \rightarrow \ldots.$$

We note that in many systems, there may be state variables where we do know the transition equation(s) (such as in an inventory problem), while there are other state variables where we do not know the transition, such as demands and prices.

9.7.3 Exogenous transitions

There are many problems where some of the state variables evolve exogenously over time: rainfall, a stock price (assuming we cannot influence the price), the travel time on a congested road network, and equipment failures. There are two ways of modeling these processes.

The first models the change in the variable. If our state variable is a price p_t, we might let $\hat{p}_{t+1}$ be the change in the price between t and $t+1$, giving us the transition function

$$p_{t+1} = p_t + \hat{p}_{t+1}.$$

This has the advantage of giving us a clean transition function that describes how the price evolves over time. With this notation, we would write $W_{t+1} = (\hat{p}_{t+1})$, so that the exogenous information is distinct from the state variable.

Alternatively, we could simply assume that the new state p_{t+1} is the exogenous information, which means we would write $W_{t+1} = p_{t+1}$. This requires that we have a process we are observing that gives us p_{t+1} without telling us how we transitioned from p_t to p_{t+1}.

9.8 THE OBJECTIVE FUNCTION

The final dimension of our model is the objective function. We divide our discussion between creating performance metrics for evaluating a decision x_t, and evaluating the policy $X^\pi(S_t)$.

9.8.1 The performance metric

Performance metrics are described using a variety of terms such as

- Rewards, profits, revenues, costs (business)
- Gains, losses (engineering)
- Strength, conductivity, diffusivity (materials science)
- Tolerance, toxicity, effectiveness (health)
- Stability, reliability (engineering)
- Risk, volatility (finance)
- Utility (economics)
- Errors (machine learning)
- Time (to complete a task)

These differ primarily in terms of units and whether we are minimizing or maximizing. These are modeled using a variety of notation systems such as c for cost, r for revenue or reward, g for gain, L or ℓ for loss, U for utility, and $\rho(X)$ as a risk measure for a random variable X.

There are many problems where there are multiple metrics. There are three strategies we can use to handle these:

1) Utility functions - We can combine different metrics into a single utility, which requires specifying weights on each metric.

2) We maximize one metric subject to constraints on the other metrics.

3) Multiobjective programming - Here we capture different objectives at the same time (such as expected profit and risk), and then let a decision-maker make an appropriate tradeoff.

Both methods (1) and (2) produce a single performance metric. These are the approaches we use in this book, since they make it possible for a computer to identify a single best decision.

9.8.2 Optimizing the policy

We close our first pass through modeling by giving the objective function for finding the best policy. Our default objective function for state-dependent problems (that is, where the contribution function and/or constraints depend on the state S_t) can be written

$$\max_{\pi\in\Pi} \mathbb{E}_{S_0}\mathbb{E}_{W_1,\ldots,W_T|S_0}\left\{\sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t))|S_0\right\}. \tag{9.20}$$

Once we get used to what we have to take the expectation over, we may just use the compact form of the expectation

$$\max_{\pi\in\Pi} \mathbb{E}\left\{\sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t))|S_0\right\}. \tag{9.21}$$

As we did in chapter 7, we write our expectation in nested form to express the possible presence of a probabilistic initial state S_0 (where we might have a distribution of belief about some information), and the observations $W_1, \ldots, W_T$. We explicitly express the dependence on S_0, even if it does not contain any probabilistic beliefs, to communicate the dependence on any static data (which could include latent variables).

The objective (9.21) is written using cumulative rewards, but there will be settings where we should use a final-reward objective. We return to this issue shortly.

9.8.3 Dependence of optimal policy on S_0

Our notation for the objective function (9.20) captures the dependence of the optimal policy on S_0 which is always present, but generally overlooked in the optimization literature. Specifically, if we find an optimal policy $X^*(S_t)$, it really should be written $X^*(S_t|S_0)$. Yes, this means that if we change the initial state, we may change the optimal policy, possibly significantly.

We already saw this in section 6.6 when we discussed tuning the stepsize policy, and then demonstrated how poorly it could work when we changed the starting point of the algorithm (this would be captured by S^0) if we did not retune the stepsize. The problem is vividly demonstrated in figure 6.10(a) when we picked a starting point in the region $[1, 2]$. In practice, reoptimizing the policy when we change S_0 can quickly become impractical. We simply make the point: reoptimizing the policy each time we change S_0 *is* impractical, but this does not mean we can pretend it is not an issue. The dependence is higly problem-dependent, but something that any algorithmic reseacher needs to be aware of.

We are going to see this issue again when designing policies in the context of stochastic lookahead models. Section 19.7.1 proposes (ahem!) that we ignore the dependence of tunable parameters on the starting state, but makes the argument that we can tolerate approximations such as this when the policy is just be used to simulate the downstream impact of a decision. Needless to say, there are a lot of unanswered questions here.

9.8.4 State-dependent variations

Depending on the setting, we might use any of the following ways of expressing our contribution function:

$$\begin{aligned}
F(x,W) &= \text{A general performance metric (to be minimized or maximized) that depends only on the decision } x \text{ and information } W \text{ that is revealed after we choose } x.\\
C(S_t,x_t) &= \text{A cost/contribution function that depends on the state } S_t \text{ and decision } x_t.\\
C(S_t,x_t,W_{t+1}) &= \text{A cost/contribution function that depends on the state } S_t \text{ and the decision } x_t, \text{ and the information } W_{t+1} \text{ that is revealed after } x_t \text{ is determined.}\\
C(S_t,x_t,S_{t+1}) &= \text{A cost/contribution function that depends on the state } S_t \text{ and the decision } x_t, \text{ after which we observe the subsequent state } S_{t+1}. \text{ This format is used in model-free settings where we do not know the transition function.}\\
C_t(S_t,x_t) &= \text{The cost/contribution function when the function itself depends on time } t.
\end{aligned}$$

We have used the notation $F(x,W)$ (as we did in chapters 5 and 7) when our problem does not depend on the state. However, as we transition to state-dependent problems, we use $C(S_t,x_t)$ (or $C(S_t,x_t,W_{t+1})$ or $C(S_t,x_t,S_{t+1})$) to communicate that the objective function (or constraints or expectation) depend on the state. Readers may choose to use any notation such as $r(\cdot)$ for reward, $g(\cdot)$ for gain, $L(\cdot)$ for loss, or $U(\cdot)$ for utility.

The state-dependent representations all depend on the state S_t (or S^n if we wish), but it is useful to say what this means. When we make a decision, we need to work with a cost function and possibly constraints where we express the dependence on S_t by writing the feasible region $\mathcal{X}_t$ as depending on t (the notation $\mathcal{X}(S_t)$ seems clumsy). For example, we might move money in a mutual fund to or from cash, buying or selling an index that is at price p_t. Let R_t be the amount of available cash, which evolves as people make deposits or withdrawals. The amount of cash could be defined by

$$R_{t+1}^{cash} = R_t^{cash} + x_t + \hat{R}_{t+1}, \tag{9.22}$$

$$R_{t+1}^{index} = R_t^{index} - x_t. \tag{9.23}$$

where $x_t > 0$ is the amount of money moved into cash by selling the index fund, while $x_t < 0$ represents money from from cash into the index fund. We have to observe the constraints

$$\begin{aligned}
x_t &\leq R_t^{index},\\
-x_t &\leq R_t^{cash}.
\end{aligned}$$

The money we make is based on what we receive from buying or selling the index fund, which we would write as

$$C(S_t,x_t) = p_t x_t,$$

where the price evolves according to the model

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \varepsilon_{t+1}.$$

For this problem, our state variable would be $S_t = (R_t, p_t, p_{t-1})$. For this example, the contribution function itself depends on the state through the prices, while the constraints (R_t^{index} and R_t^{cash}) also vary dynamically and are part of the state.

Now imagine that we have to make the decision to buy or sell shares of our index fund, but the price we get is based on the closing price, which is not known when we make our decision. In this case, we would write our contribution function as

$$C(S_t, x_t, W_{t+1}) = p_{t+1}x_t,$$

where $W_{t+1} = \hat{p}_{t+1} = p_{t+1} - p_t$. We note that our policy $X^\pi(S_t)$ for making the decision x_t is not allowed to use W_{t+1}; rather, we have to wait until time $t+1$ before evaluating the quality of the decision.

Finally, consider a model of a hydroelectric reservoir where we have to manage the inventory in the reservoir, but where the dynamics describing its evolution is much more complicated than equations such as (9.22) and (9.23). In this setting, we can observe the reservoir level R_t, then make a decision of how much water to release out of the reservoir x_t, after which we observe the updated reservoir level R_{t+1}. This is similar to observing an updated price p_{t+1}. For these problems, we might let W_{t+1} be the new state, in which case our "transition equations" are just

$$S_{t+1} = W_{t+1}.$$

Alternatively, we may find it more natural to write the contribution function $C(S_t, x_t, S_{t+1})$, which is fairly common, but there are settings where we have transition equations for some variables but not others.

We use $C(S_t, x_t)$ as our standard notation (in some settings we will index the contribution function by time, as in $C_t(S_t, x_t)$). If we find ourselves writing the contribution in a form that needs $C(S_t, x_t, W_{t+1})$ as we illustrated above, we can always break the contribution into the parts that can be computed at time t, and the parts that cannot be computed until time $t+1$. We can easily write this as

$$C_t(S_t, x_t, W_{t+1}) = C_t^1(S_t, x_t) + C_{t+1}^2(S_t, x_t, W_{t+1}).$$

where $C_t^1(S_t, x_t) = -cx_t$ captures the components of the contribution function that can be computed at time t, and $C_{t+1}^2(S_t, x_t, W_{t+1}) = p\min\{S_t + x_t, W_{t+1}\}$ captures the components that cannot be computed until time $t+1$.

Next create the contribution function

$$\tilde{C}_t(S_t, x_t) = C_t^2(S_{t-1}, x_{t-1}, W_t) + C_t^1(S_t, x_t).$$

Now optimize the sum of contribution functions $\tilde{C}_t(S_t, x_t)$ over the horizon. This strategy may seem unintuitive (or unappealing) since $C_{t-1}^2(S_{t-1}, x_{t-1}, W_t)$ does not depend on x_t, and we are not capturing the impact of x_t on revenue. However, these are simply cosmetic issues. Simply moving the contributions that depend on W_{t+1} to the next time period will not change the overall performance of any optimizing policy that we propose in chapter 11 (or develop in the rest of the book).

9.8.5 Uncertainty operators

An important issue when optimizing under uncertainty is that we have to decide how to evaluate the distribution of the objective function for a policy. Some choices we can use are:

- The expectation operator $\mathbb{E}\{\cdot|S_0\}$ - We use this as our default operator, since it is easily the one that is most commonly used.
- The risk operator $\rho(\cdot)$ - This is actually a family of operators that are designed to capture the tails or spread of the distribution of outcomes. Some examples are:
  - Value at risk $F^\pi_\alpha = VaR_\alpha(F^\pi)$ - This is the value F^π_α of the α-quantile of a random variable F^π giving the performance of the policy $X^\pi(S)$. If we are maximizing, we might use the 10^{th} percentile to protect ourselves from doing poorly.
  - Conditional value at risk $CVaR_\alpha(Z)$ - Also known as the average value at risk or expected shortfall, this is the expectation of $Z = \max\{0, F_\alpha - F^\pi\}$ (if we are maximizing).
  - There is a host of potential other measures, such as the worst performance over the horizon, the α-percentile over all the time periods, and so on.
- Robust optimization, where we would use the worst possible outcome which we can write

$$\min_{\omega\in\Omega} F^\pi(\omega),$$

  where $F^\pi(\omega)$ is the performance of the policy for sample path ω. This means that our optimization problem is

$$\max_\pi \min_{\omega\in\Omega} F^\pi(\omega).$$

Our default operator is the expectation, which is often used even when a risk measure is used in a stochastic lookahead model. For example, there is a substantial community called "robust optimization" (see section 2.1.14) which might use a stochastic lookahead policy with a robust objective, but which then evaluates the "robust" policy by simulating it many times and taking an average (which means using an expectation to evaluate the policy). We revisit this in chapter 19.

9.9 ILLUSTRATION: AN ENERGY STORAGE MODEL

In section 9.1 we presented a very simple energy storage problem where we have to determine when to buy energy from the grid, or sell it back to the grid. We are going to expand on this model, first by introducing the ability to draw energy from the grid or a wind farm which is stored in a battery, from which we draw energy to meet a demand D_t. Then, we are going to make the price process into a simple first-order process.

The decision variables are given by

x^G_t = the energy we purchase from the grid ($x^G_t > 0$) or sell back to the grid ($x^G_t < 0$) which moves to or from the battery,
x^E_t = the energy generated from a wind farm at time t to the battery,
x^D_t = the energy moved from the battery to meet the demand D_t.

We then define the exogenous inputs

E_t = the energy available from the wind farm at time t,
D_t = the demand for energy at time t.

The flows have to satisfy the constraints

$$\begin{aligned} x_t^E &\leq E_t, && (9.24)\\ x_t^G + x_t^E &\leq R^{max} - R_t, && (9.25)\\ x_t^D &\leq R_t, && (9.26)\\ x_t^D &\leq D_t, && (9.27)\\ -x_t^G &\leq R_t. && (9.28) \end{aligned}$$

Equation (9.24) limits the energy we store in the battery from the wind farm to the available wind in the wind farm. Equation (9.25) limits the total energy from the grid and the wind farm to the available capacity in the battery. Equation (9.26) limits the amount we use from the battery to serve the demand to the amount in the battery, while equation (9.27) limits the energy sent to meet the demand to the demand itself. Equation (9.28) limits the amount of energy sent back to the grid (this is where $x_t^G < 0$) to the amount in the battery.

The transition equations are given by

$$\begin{aligned} R_{t+1} &= R_t + x_t,\\ p_{t+1} &= p_t + \varepsilon_{t+1}, \end{aligned}$$

where $\varepsilon_{t+1} \sim N(0, \sigma^2)$ (before we had assumed that we just observed p_{t+1}). We assume that the changes in prices $\hat{p}_t$ are independent across time. We assume that the energy E_t from the wind farm and the demand D_t is observed without models of their evolution. Below we address some modeling issues related to forecasting E_t.

For this basic system, the state variable would be

$$S_t = \big((R_t, E_t, D_t), p_t\big).$$

We are now going to step through a series of variations where we modify the price process, and then describe the effect of the change on the state variable.

9.9.1 With a time-series price model

We begin by replacing our simple price process in equation (9.29) with a time series model given by

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}. \quad (9.29)$$

It is surprisingly common for people to say that p_t is the "state" of the price process, and then insist that it is no longer Markovian (it would be called "history dependent"), but "it can be made Markovian by expanding the state variable," which would be done by including p_{t-1} and p_{t-2}. Using our definition, the state is all the information needed to model the process from time t onward, which means that the state of our price process is (p_t, p_{t-1}, p_{t-2}). This means our system state variable is now

$$S_t = \big((R_t, E_t, D_t), (p_t, p_{t-1}, p_{t-2})\big).$$

We then have to modify our transition function so that the "price state variable" at time $t+1$ becomes (p_{t+1}, p_t, p_{t-1}).

9.9.2 With passive learning

The price model in equation (9.29) assumed the coefficients $\theta = (\theta_0, \theta_1, \theta_2)$ were known. Now assume that the coefficients are unknown and have to be learned along the way, as in

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon_{t+1}. \tag{9.30}$$

Here, we have to recursively update our estimate $\bar{\theta}_t$ which we can do using recursive least squares which we introduced in section 3.8. To do this, let

$$\begin{aligned} \bar{p}_t &= (p_t, p_{t-2}, p_{t-2})^T, \\ \bar{F}_t(\bar{p}_t|\bar{\theta}_t) &= (\bar{p}_t)^T \bar{\theta}_t. \end{aligned}$$

The updating equations for $\bar{\theta}_t$ are given by

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \frac{1}{\gamma_t} M_t \bar{p}_t \varepsilon_{t+1}, \tag{9.31}$$

$$\varepsilon_{t+1} = \bar{F}_t(\bar{p}_t|\bar{\theta}_t) - p_{t+1}, \tag{9.32}$$

$$M_{t+1} = M_t - \frac{1}{\gamma_t} M_t (\bar{p}_t)(\bar{p}_t)^T M_t, \tag{9.33}$$

$$\gamma_t = 1 - (\bar{p}_t)^T M_t \bar{p}_t. \tag{9.34}$$

To compute these equations, we need the three-element vector $\bar{\theta}_t$ and the 3×3 matrix M_t. These then need to be added to our state variable, giving us

$$S_t = \big((R_t, E_t, D_t), (p_t, p_{t-1}, p_{t-2}), (\bar{\theta}_t, M_t)\big),$$

which has 18 continuous dimensions. We then have to include equations (9.31) - (9.34) in our transition function.

9.9.3 With active learning

There are many settings where the decisions we make either directly affect or at least influence what we observe. We are going to assume that our decision x_t^{GB} to buy or sell energy from or to the grid can have an impact on prices. We might propose a modified price model given by

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \bar{\theta}_{t3} x_t^{GB} + \varepsilon_{t+1}. \tag{9.35}$$

Now, buying or selling large quantities from or to the grid can push prices higher or lower, allowing us to explore different regions of the model. This is known as *active learning*, a topic we introduced in chapter 7 for both offline and online settings.

This change in our price model does not affect the state variable from the previous model, aside from adding one more element to $\bar{\theta}_t$, with the required changes to the matrix M_t. The change will, however, have an impact on the policy. It is easier to learn θ_{t3} by varying x_t^{GB} over a wide range, which means trying values of x_t^{GB} that do not appear to be optimal given our current estimate of the vector $\bar{\theta}_t$. Making decisions partly just to learn (to make better decisions in the future) is the essence of *active learning*, best known in the field of multiarmed bandit problems.

9.9.4 With rolling forecasts

Forecasting is such a routine activity in operational problems, it may come as a surprise that we have been modelling these problems incorrectly.

Assume we have a forecast $f^E_{t,t+1}$ of the energy E_{t+1} from the wind farm, which means

$$E_{t+1} = f^E_{t,t+1} + \varepsilon_{t+1,1}, \tag{9.36}$$

where $\varepsilon_{t+1,1} \sim N(0, \sigma^2_\varepsilon)$ is the random variable capturing the one-period-ahead error in the forecast.

Equation (9.36) introduces a new variable, the forecast $f^E_{t,t+1}$, which must now be added to the state variable. This means we now need a transition equation to describe how $f^E_{t,t+1}$ evolves over time. We do this by using a two-period-ahead forecast, $f^E_{t,t+2}$, which is basically a forecast of $f^E_{t+1,t+2}$, plus an error, giving us

$$f^E_{t+1,t+2} = f^E_{t,t+2} + \varepsilon_{t+1,2}, \tag{9.37}$$

where $\varepsilon_{t+1,2} \sim N(0, \sigma^2_\varepsilon)$ is the two-period-ahead error (we are assuming that the variance in a forecast increases linearly with time). Now we have to put $f^E_{t,t+2}$ in the state variable, which generates a new transition equation. This generalizes to

$$f^E_{t+1,t'} = f^E_{t,t'} + \varepsilon_{t+1,t'-t}, \tag{9.38}$$

where $\varepsilon_{t+1,t'-t} \sim N(0, \sigma^2_\varepsilon)$.

This stops, of course, when we hit the planning horizon H. This means that we now have to add

$$f^E_t = (f^E_{tt'})^{t+H}_{t'=t+1}$$

to the state variable, with the transition equations (9.38) for $t' = t+1, \ldots, t+H$. Combined with the learning statistics, our state variable is now

$$S_t = \big((R_t, E_t, D_t), (p_t, p_{t-1}, p_{t-2}), (\bar{\theta}_t, M_t), f^E_t\big).$$

It is useful to note that we have a nice illustration of the three elements of our state variable:

(R_t, E_t, D_t)	=	the physical state variables (energy in the battery, energy available from the wind farm, current demand for energy),
(p_t, p_{t-1}, p_{t-2})	=	other information (recent prices),
$((\bar{\theta}_t, M_t), f^E_t)$	=	the belief state, since these parameters determine the distribution of belief about variables that are not known perfectly.

This state variable has 42 dimensions: three for the physical states, three for prices, 12 for the endogenous forecasts, and 24 for the rolling forecasts.

9.10 BASE MODELS AND LOOKAHEAD MODELS

There is a subtle but critical distinction between a "model" of a real problem, and what we will come to know as a "lookahead model," which is an approximation which is used to

peek into the future (typically with various convenient approximations) for the purpose of making a decision now. We are going to describe lookahead models in far greater depth in chapter 19, but we feel that it is useful to make the distinction now.

Using the framework presented in this chapter, we can write almost any sequential decision process in the compact form

$$max_{\pi\in\Pi}\mathbb{E}\left\{\sum_{t=0}^{T}C_t(S_t,X_t^\pi(S_t))|S_0\right\}, \tag{9.39}$$

where $S_{t+1} = S^M(S_t, X_t^\pi(S_t), W_{t+1})$. Of course, we have to specify our model for $(W_t)_{t=0}^{T+1}$ in addition to defining the state variable (later we will address the issue of identify our class of policies).

For the moment, we view (9.39) (along with the transition function) as "the problem" that we are trying to solve. If we find an effective policy, we assume we have solved "the problem." However, we are going to learn that in dynamic systems, we are often solving a problem at some time t over a horizon $(t, \ldots, t + H)$, where we simply set $t = 0$ and number time periods accordingly. The question is: are we interested in the solution over the entire planning horizon, or just the decision in the first time period?

Given the widespread use of lookahead models, we need a term to identify when we are presenting a model of a problem we wish to solve. We might use the term "real model" to communicate that this is our model of the real world. Statisticians use the term "true model," but this seems to assume that we have somehow perfectly modeled a real problem, which is never the case. Some authors use the term "nominal model," but we feel that this is not sufficiently descriptive.

In this book, we use the term *base model* since we feel that this communicates the idea that this is the model we wish to solve. We take the position that regardless of any modeling approximations that have been introduced (either for reasons of tractability or availability of data), this is "the" model we are trying to solve.

Later, we are going to introduce approximations of our base model, which may still be quite difficult to solve. Most important will be the use of lookahead models, which we discuss in depth in chapter 19.

9.11 A CLASSIFICATION OF PROBLEMS*

It is useful to contrast problems based on two key dimensions: First, whether the objective function is final-reward or cumulative-reward, and second, whether the objective function is state-independent (learning problems, which we covered in chapters 5 and 7) or state-dependent (traditional dynamic programs), which we began treating in chapter 8, and which will be the focus of the remaining chapters.

This produces four problem classes which are depicted in table 9.3. We have numbered the classes in increasing order of complexity, with the warning that class 4 is particularly difficult to parse. In this section, we are going to write out the objectives in expectation form, but in the section that follows, we are going to show how to simulate the expectations, which we feel will make the expectations easier to understand. It may help to flip forward to section 9.12 to peek at the simulated version of each expression.

Class 1) State-independent, final reward - This describes classical search problems where we are trying to find the best algorithm (which we call a policy π) for finding the best solution $x^{\pi,N}$ within our budget N. After n experiments the state S^n captures

	Offline Final 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 9.3 Comparison of formulations for state-independent (learning) vs. state-dependent problems, and offline (final reward) and online (cumulative reward).

only our belief state about the function $\mathbb{E}F(x,W)$, and our decisions are made with a policy (or algorithm) $x^n = X^\pi(S^n)$. We can write this problem as

$$\max_\pi \mathbb{E}\{F(x^{\pi,N},\widehat{W})|S^0\} = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\mathbb{E}_{\widehat{W}|S^0}F(x^{\pi,N},\widehat{W}), \tag{9.40}$$

where $W^1,\ldots,W^N$ are the observations of W while learning the function $\mathbb{E}F(x,W)$, and $\widehat{W}$ is the random variable used for testing the final design $x^{\pi,N}$. The distinguishing characteristics of this problem are a) that the function $F(x,W)$ depends only on x and W, and not on the state S^n, and b) that we evaluate our policy $X^\pi(S)$ only after we have exhausted our budget of N experiments. We do allow the function $F(x,W)$, the observations $W^1,\ldots,W^N$ and the random variable $\widehat{W}$ to depend on the initial state S_0, which includes any deterministic parameters, as well as probabilistic information (such as a Bayesian prior) that describes any unknown parameters (such as how the market responds to price).

Class 2) State-independent, cumulative reward - Here we are looking for the best policy that learns while it optimizes. This means that we are trying to maximize the sum of the rewards received within our budget. This is the classic multiarmed bandit problem that we first saw in chapter 7 if the decisions x were discrete and we did not have access to derivatives (but we are not insisting on these limitations). We can write the problem as

$$\max_\pi \mathbb{E}\left\{\sum_{n=0}^{N-1} F(X^\pi(S^n),W^{n+1})|S^0\right\} = \mathbb{E}_{S^0}\mathbb{E}_{W^1,\ldots,W^N|S^0}\sum_{n=0}^{N-1} F(X^\pi(S^n),W^{n+1}). \tag{9.41}$$

Class 3) State-dependent, cumulative reward - We now transition to problems where we are maximizing contributions that depend on the state variable, the decision, and possibly (but not always) random information that arrives after we make a decision (if it arrived before, it would be included in the state variable). For this reason, we are going to switch from our notation $F(x,W)$ to our notation $C(S,x,W)$ (or, in a time-indexed environment, $C(S_t,x_t,W_{t+1})$). As with the multiarmed bandit

problem (or more generally, Class (2) problems), we want to find a policy that learns while implementing. These problems can be written

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\} = \mathbb{E}_{S_0}\mathbb{E}_{W_1,\ldots,W_T|S_0}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t), W_{t+1})|S_0\right\}. \tag{9.42}$$

State variables in this problem class may include any of the following:

- Variables that are controlled (or influenced) by decisions (such as inventory or the location of a sensor on a graph). These variables directly affect the contribution function (such as price) or the constraints (such as the inventory).
- Variables that evolve exogenously (such as the wind speed or price of an asset).
- Variables that capture our belief about a parameter that are only used by the policy.

When we consider that our state S_t may include a controllable physical state R_t, exogenous information I_t and/or a belief state B_t, we see that this covers a very broad range of problems. The key feature here is that our policy has to maximize cumulative contributions as we progress, which may include learning (if there is a belief state).

Class 4) State-dependent, final reward - For our state-independent function $F(x, W)$ we were looking for the best policy to learn the decision $x^{\pi,N}$ to be implemented. In this setting, we can think of the policy as a *learning policy*, while $x^{\pi,N}$ is the *implementation decision*. In the state-dependent case, the implementation decision becomes one that depends on the state (at least, part of the state), which is a function we call the *implementation policy*. We designate the implementation policy by $X^{\pi^{imp}}(S|\theta^{imp})$, which we write as depending on a set of parameters θ^{imp} which have to be learned. We designate the learning policy for learning θ^{imp} by $\Theta^{\pi^{lrn}}(S|\theta^{lrn})$ which proceeds by giving us parameters $\theta^{imp,n} = \Theta^{\pi^{lrn}}(S^n|\theta^{lrn})$. The problem can be written

$$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{imp}}(S|\theta^{imp}), \widehat{W})|S^0\} = \mathbb{E}_{S^0}\mathbb{E}^{\pi^{lrn}}_{W^1,\ldots,W^N|S^0}\mathbb{E}^{\pi^{imp}}_{S|S^0}\mathbb{E}_{\widehat{W}|S^0} C(S, X^{\pi^{imp}}(S|\theta^{imp}), \widehat{W}). \tag{9.43}$$

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. We use the expectation operator $\mathbb{E}^{\pi^{lrn}}$ indexed by the learning policy when the expectation is over a random variable whose distribution is affected by the learning policy.

The learning policy could be a stochastic gradient algorithm to learn the parameters θ^{imp}, or it could be one of our derivative-free methods such as interval estimation or upper confidence bounding. The learning policy could be algorithms for learning value functions such as Q-learning (see equations (2.19)-(2.21) in chapter 2), or the parameters of any of the derivative-free search algorithms in chapter 7.

We typically cannot compute the expectation $\mathbb{E}_S^{\pi^{imp}}$ since it depends on the implementation policy which in turn depends on the learning policy. As an alternative, we can run a simulation over a horizon $t = 0, \ldots, T$ and then divide by T to get an average contribution per unit time. This simulation is performed using our testing random variable $\widehat{W}_t$, since we are evaluating the policy after we have learned the implementation policy. Let $\widehat{W}^n = (\widehat{W}_1^n, \ldots, \widehat{W}_T^n)$ be a simulation over our horizon. This allows us to write our learning problem as

$$\max_{\pi^{lrn}} \mathbb{E}_{S^0} \mathbb{E}^{\pi^{imp}}_{(W_t^n)_{t=0}^T)_{n=0}^N | S^0} \left(\mathbb{E}^{\pi^{imp}}_{(\widehat{W}_t)_{t=0}^T | S^0} \frac{1}{T} \sum_{t=0}^{T-1} C(S_t, X^{\pi^{imp}}(S_t|\theta^{imp}), \widehat{W}_{t+1}) \right). \tag{9.44}$$

This parallels class (1) problems. We are searching over learning policies which determines the implementation policy through $\theta^{imp} = \Theta^{\pi^{lrn}}(S|\theta^{lrn})$, where the simulation over time replaces $F(x, W)$ in the state-independent formulation. The sequence $(W_t^n)_{t=0}^T, n = 1, \ldots, N$ 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 $(\widehat{W}_t)_{t=0}^T$, where we again assume we start our simulations at $S^0 = S_0$.

9.12 POLICY EVALUATION*

While it is certainly useful to characterize these four problem classes, it is an entirely different matter to compute the expectations in equations (9.40)-(9.44). The best way to approach this task (in fact, the best way to actually understand the expectations) is to simulate them. In this section we describe how to approximate each expectation using simulation.

We begin by fixing a policy $X^\pi(S_t|\theta)$ parameterized by some vector θ, which can be anything including a learning policy such as Thompson sampling, a stochastic gradient algorithm with a particular stepsize policy, or a direct lookahead policy. In problem class (1), $X^\pi(S_t|\theta)$ is a pure learning policy which learns an implementation decision $x^{\pi,N}(\theta)$. In classes (2) and (3), it is a policy where we learn as we implement. In class (4), we use a learning policy $\Theta^{\pi^{lrn}}(S_t|\theta^{lrn})$ to learn the parameter θ^{imp} of an implementation policy $X^{\pi^{imp}}(S_t|\theta^{imp})$ where $\theta^{imp} = \Theta^{\pi^{lrn}}(\theta^{lrn})$ depends on the learning policy.

Throughout, we are going to use θ (or θ^{lrn}) as a (possibly vector-valued) parameter that controls our learning policy (for classes (1) and (4)) or the implementation policy (possibly with learning) for classes (2) and (3). The vector θ (or θ^{lrn}) might be the parameters governing the behavior of any adaptive learning algorithm.

We now need to evaluate how well this policy works. We start with state S^0 if we are in problem classes (1) or (4), or S_0 if we are in problem class (3), and S^0 or S_0 if we are in problem class (2). From the initial state, we pick initial values of any parameters, either because they are fixed, or by drawing them from an assumed distribution (that is, a Bayesian prior).

We next address the process of simulating a policy for each of the four problem classes.

Class 1) State-independent, final reward - From an initial state S^0, we use our (learning) policy to make decision $x^0 = X^\pi(S^0|\theta)$, and then observe outcome W^1, producing

an updated state S^1 (in this problem, S^n is a pure knowledge state). The parameter θ controls the behavior of our learning policy. We repeat this until our budget is depleted, during which we observe the sequence $W^1(\omega), \ldots, W^N(\omega)$, where we let ω represent a particular sample path. At the end we learn state S^N, from which we find our best solution (the final design) $x^{\pi,N}$, which we write as $x^{\pi,N}(\theta|\omega)$ to express its dependence on the learning policy π (parameterized by θ) and the sample path ω.

We then evaluate $x^{\pi,N}(\theta|\omega)$ by simulating $\mathbb{E}_{\widehat{W}} F(x^{\pi,N}(\theta|\omega), \widehat{W})$ by repeatedly sampling from $\widehat{W}$ to get sampled estimates of $\mathbb{E}_{\widehat{W}} F(x^{\pi,N}(\theta|\omega), \widehat{W})$. Let $\widehat{W}(\psi)$ be a particular realization of $\widehat{W}$. A sampled estimate of the policy π (which we assume is parameterized by θ) is given by

$$F^\pi(\theta|\omega, \psi) = F(x^{\pi,N}(\theta|\omega), \widehat{W}(\psi)). \tag{9.45}$$

We now average over a set of K samples of ω, and L samples of ψ, giving us

$$\bar{F}^\pi(\theta) = \frac{1}{K}\frac{1}{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L} F^\pi(\theta|\omega^k, \psi^\ell). \tag{9.46}$$

Class 2) State-independent, cumulative reward - This problem can be interpreted in two ways. As the cumulative reward version of problem class (1), we simulate our policy for N iterations, giving us the sequence $(S^0, x^0, W^1, \ldots, x^{N-1}, W^N, S^N)$. Here, we accumulate our rewards, producing a sampled estimate

$$F^\pi(\theta|\omega) = \sum_{n=0}^{N-1} F(X^\pi(S^n|\theta), W^{n+1}(\omega)). \tag{9.47}$$

Unlike class (1), we evaluate our policy as we go, avoiding the need for the final step at the end. We would then compute an average using

$$\bar{F}^\pi(\theta) = \frac{1}{K}\sum_{k=1}^{K} F^\pi(\theta|\omega^k), \tag{9.48}$$

over a sample of K observations.

We can also recast this problem as simulating over time, where we just replace W^n with W_t and S^n with S_t.

Class 3) State-dependent, cumulative reward - This is the state-dependent version of problem class (2), which we model as evolving over time. Starting in state S_0, we simulate the policy much as we did in equation (9.47) which is given by

$$F^\pi(\theta|\omega) = \sum_{t=0}^{T-1} C(S_t(\omega), X^\pi(S_t(\omega)|\theta), W_{t+1}(\omega)). \tag{9.49}$$

We then average over sample paths to obtain

$$\bar{F}^\pi(\theta) = \frac{1}{K}\sum_{k=1}^{K} F^\pi(\theta|\omega^k). \tag{9.50}$$

Class 4) State-dependent, final reward - We now have a hybrid of problem classes (1) and (3), where we use a learning policy $\Theta^{\pi^{lrn}}(S|\theta^{lrn})$ to learn the parameters of an implementation policy $X^{\pi^{imp}}(S_t|\theta^{imp})$, where the parameter $\theta^{imp} = \Theta^{\pi^{lrn}}(\theta^{lrn})$ that determines the behavior of the implementation policy depends on the learning policy π^{lrn} and its tunable parameters θ^{lrn}. We then have to evaluate the implementation policy, just as we evaluated the final design $x^{\pi,N}(\theta)$ in class (1), where $x^{\pi,N}(\theta)$ is the implementation decision that depends on the learning policy π and its parameters θ.

In class (1), we evaluated the implementation decision $x^{\pi,N}(\theta)$ by simulating $\widehat{W}$ to obtain estimates of $F(x^{\pi,N}, \widehat{W})$. Now we have to take an expectation over the state S which we do by simulating our implementation policy $X^{\pi^{imp}}(S_t|\theta^{imp})$ starting in state S_0 until the end of our horizon S_T. One simulation from 0 to T is comparable to an evaluation of $F(x, W)$. This means that a sample path ω, which in (1) was one observation of $W_1, \ldots, W_T$, is an observation of $(W_t^n, t = 1, \ldots, T), n = 0, \ldots, N$. This observation then produces the implementation policy $X^{\pi^{imp}}(S_t|\theta^{imp})$ (whereas in class (1) problems it produced the implementation decision $x^{\pi,N}(\theta|\omega)$).

To simulate the value of the policy, we simulate one last set of observations $\widehat{W}_1(\psi), \ldots, \widehat{W}_T(\psi)$ which, combined with our implementation policy which we write as $X^{\pi^{imp},N}(S_t|\theta^{imp}, \omega)$ produces a sequence of states $S_t(\psi)$, giving us the estimate

$$F^{\pi}(\theta^{lrn}|\omega, \psi) = \frac{1}{T}\sum_{t=0}^{T} C(S_t(\psi), X^{\pi^{imp}}(S_t(\psi)|\theta^{imp}, \omega), \widehat{W}_{t+1}(\psi)), \quad (9.51)$$

where we need to remember that $\theta^{imp} = \Theta^{\pi^{lrn}}(\theta^{lrn})$. We finally average over a set of K samples of ω, and L samples of ψ, giving us

$$\bar{F}^{\pi}(\theta^{lrn}) = \frac{1}{K}\frac{1}{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L} F^{\pi}(\theta^{lrn}|\omega^k, \psi^\ell), \quad (9.52)$$

We now have a way of computing the performance of a policy $\bar{F}^{\pi}(\theta)$, which may be a learning policy for classes (1) and (4), or an implementation (and learning) policy for classes (2) and (3).

9.13 ADVANCED PROBABILISTIC MODELING CONCEPTS**

Sequential decision problems introduce some very subtle issues when bridging with classical probability theory. This material is not important for readers who just want to focus on models and algorithms. However, understanding how the probability community thinks of stochastic dynamic programs provides a fresh perspective that brings a deep pool of theory from the probability community.

Section 9.13.1 provides a beginners introduction to what is known as a measure-theoretic view of information, which provides some basic concepts that are used throughout advanced research papers in stochastic optimization. Then, section 9.13.2 provides a short primer of terms that are widely used throughout stochastic optimization papers which represent what is arguably the most common uses of the measure-theoretic terminology presented in section 9.13.1. We emphasize that while these concepts are widely used in the mathematical research literature, they are not necessary for modeling and solving real problems.

9.13.1 A measure-theoretic view of information**

For readers interested in proving theorems or reading theoretical research articles, it is useful to have a more fundamental understanding of information.

When we work with random information processes and uncertainty, it is standard in the probability community to define a probability space, which consists of three elements. The first is the set of outcomes Ω, which is generally assumed to represent all possible outcomes of the information process (actually, Ω can include outcomes that can never happen). If these outcomes are discrete, then all we would need is the probability of each outcome $p(\omega)$.

It is nice to have a terminology that allows for continuous quantities. We want to define the probabilities of our events, but if ω is continuous, we cannot talk about the probability of an outcome ω. However we can talk about a set of outcomes $\mathcal{E}$ that represent some specific event (if our information is a price, the event $\mathcal{E}$ could be all the prices that constitute the event that the price is greater than some number). In this case, we can define the probability of an outcome $\mathcal{E}$ by integrating the density function $p(\omega)$ over all ω in the event $\mathcal{E}$.

Probabilists handle continuous outcomes by defining a set of events $\mathfrak{F}$, which is literally a "set of sets" because each element in $\mathfrak{F}$ is itself a set of outcomes in Ω. This is the reason we resort to the script font $\mathfrak{F}$ as opposed to our calligraphic font for sets; it is easy to read $\mathcal{E}$ as "calligraphic E" and $\mathfrak{F}$ as "script F." The set $\mathfrak{F}$ has the property that if an event $\mathcal{E}$ is in $\mathfrak{F}$, then its complement $\Omega \setminus \mathcal{E}$ is in $\mathfrak{F}$, and the union of any two events $\mathcal{E}_X \cup \mathcal{E}_Y$ in $\mathfrak{F}$ is also in $\mathfrak{F}$.

$\mathfrak{F}$ is called a "sigma-algebra" (which may be written "σ-algebra"), and is a countable union of events in Ω. An understanding of sigma-algebras is not important for computational work, but can be useful in certain types of proofs (the proof in section 5.10.3 is a good example). Sigma-algebras are without question one of the more arcane devices used by the probability community, but once they are mastered, they are a powerful theoretical tool (but useless for modeling or computation, which is the reason why we do not use them elsewhere).

Finally, it is required that we specify a probability measure denoted $\mathcal{P}$, which gives the probability (or density) of an outcome ω which can then be used to compute the probability of an event in $\mathfrak{F}$.

We can now define a formal probability space for our exogenous information process as $(\Omega, \mathfrak{F}, \mathcal{P})$, sometimes known as the "holy trinity" in probability. If we wish to take an expectation of some quantity that depends on the information, say $Ef(W)$, then we would sum (or integrate) over the set $\mathcal{E} \in \mathfrak{F}$ multiplied by the probability (or density) $\mathcal{P}$.

This notation is especially powerful for "static" problems where there are two points in time: before we see the random variable W, and after. This creates a challenge when we have sequential problems where information evolves over time. Probabilists have adapted the original concept of probability spaces $(\Omega, \mathfrak{F}, \mathcal{P})$ by manipulating the set of events $\mathfrak{F}$, as we show next.

It is important to emphasize that ω represents *all* the information that will become available, over all time periods. As a rule, we are solving a problem at time t, which means we do not have the information that will become available after time t. To handle this, we let $\mathfrak{F}_t$ be the sigma-algebra representing events that can be created using only the information up to time t. To illustrate, consider an information process W_t consisting of a single 0 or 1 in each time period. W_t may be the information that a customer purchases a jet aircraft, or the event that an expensive component in an electrical network fails. If we look over three time periods, there are eight possible outcomes, as shown in table 9.4.

Outcome	Time period		
ω	1	2	3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Table 9.4 Set of demand outcomes

Let $\mathcal{E}_{\{W_1\}}$ be the set of outcomes ω that satisfy some logical condition on W_1. If we are at time $t = 1$, we only see W_1. The event $W_1 = 0$ would be written

$$\mathcal{E}_{\{W_1=0\}} = \{\omega | W_1 = 0\} = \{1, 2, 3, 4\}.$$

The sigma-algebra $\mathfrak{F}_1$ would consist of the events

$$\{\mathcal{E}_{\{W_1=0\}}, \mathcal{E}_{\{W_1=1\}}, \mathcal{E}_{\{W_1 \in \{0,1\}\}}, \mathcal{E}_{\{W_1 \notin \{0,1\}\}}\}.$$

Now assume that we are at time $t = 2$ and have access to W_1 and W_2. With this information, we are able to divide our outcomes Ω into finer subsets. Our history H_2 consists of the elementary events $\mathcal{H}_2 = \{(0,0), (0,1), (1,0), (1,1)\}$. Let $h_2 = (0,1)$ be an element of H_2. The event $\mathcal{E}_{\{h_2=(0,1)\}} = \{3, 4\}$. At time $t = 1$, we could not tell the difference between outcomes 1, 2, 3, and 4; now that we are at time 2, we can differentiate between $\omega \in \{1, 2\}$ and $\omega \in \{3, 4\}$. The sigma-algebra $\mathfrak{F}_2$ consists of all the events $\mathcal{E}_{h_2}, h_2 \in \mathcal{H}_2$, along with all possible unions and complements.

Another event in $\mathfrak{F}_2$ is $\{\omega | (W_1, W_2) = (0,0)\} = \{1, 2\}$. A third event in $\mathfrak{F}_2$ is the union of these two events, which consists of $\omega = \{1, 2, 3, 4\}$ which, of course, is one of the events in $\mathfrak{F}_1$. In fact, every event in $\mathfrak{F}_1$ is an event in $\mathfrak{F}_2$, but not the other way around. The reason is that the additional information from the second time period allows us to divide $\mathfrak{F}$ into finer set of subsets. Since $\mathfrak{F}_2$ consists of all unions (and complements), we can always take the union of events, which is the same as ignoring a piece of information.

By contrast, we cannot divide $\mathfrak{F}_1$ into a finer subsets. The extra information in $\mathfrak{F}_2$ allows us to filter Ω into a finer set of subsets than was possible when we only had the information through the first time period. If we are in time period 3, $\mathfrak{F}$ will consist of each of the individual elements in Ω as well as all the unions needed to create the same events in $\mathfrak{F}_2$ and $\mathfrak{F}_1$.

From this example, we see that more information (that is, the ability to see more elements of $W_1, W_2, \ldots$) allows us to divide Ω into finer-grained subsets. For this reason, we can always write $\mathfrak{F}_{t-1} \subseteq \mathfrak{F}_t$. $\mathfrak{F}_t$ always consists of every event in $\mathfrak{F}_{t-1}$ in addition to other finer events. As a result of this property, $\mathfrak{F}_t$ is termed a *filtration*. It is because of this

interpretation that the sigma-algebras are typically represented using the script letter F (which literally stands for filtration) rather the more natural letter H (which stands for history). The fancy font used to denote a sigma-algebra is used to designate that it is a set of sets (rather than just a set).

It is *always* assumed that information processes satisfy $\mathfrak{F}_{t-1} \subseteq \mathfrak{F}_t$. Interestingly, this is not always the case in practice. The property that information forms a filtration requires that we never "forget" anything. In real applications, this is not always true. Assume, for example, that we are doing forecasting using a moving average. This means that our forecast f_t might be written as $f_t = (1/T)\sum_{t'=1}^{T} \hat{D}_{t-t'}$. Such a forecasting process "forgets" information that is older than T time periods.

By far the most widespread use of the notation $\mathfrak{F}_t$ is to represent the information we know at time t. For example, let W_{t+1} be the information that we will learn at time $t+1$. If we are sitting at time t, we might use a forecast $f^W_{t,t+1}$ which would be written

$$f^W_{t,t+1} = \mathbb{E}\{W_{t+1}|\mathfrak{F}_t\}. \tag{9.53}$$

Conditioning on $\mathfrak{F}_t$ means conditioning on what we know at time t which some authors will write as

$$f^W_{t,t+1} = \mathbb{E}_t W_{t+1}. \tag{9.54}$$

Equations (9.53) and (9.54) are equivalent, and both would be read "the conditional expectation of W_{t+1} given what we know at time t."

If we do not include this conditioning, then this is the same as an expectation we would make at time 0, which we could write

$$\begin{aligned} f^W_{0,t+1} &= \mathbb{E}W_{t+1} \\ &= \mathbb{E}\{W_{t+1}|\mathfrak{F}_0\}. \end{aligned}$$

There are numerous textbooks on measure theory. For a nice introduction to measure-theoretic thinking (and in particular the value of measure-theoretic thinking), see Pollard (2002) for an introduction to measure-theoretic probability, or the advanced text Cinlar (2011). For an illustration of mathematics using this notation, see the "More modern proof" of convergence for stochastic gradient algorithms in section 5.10.3.

9.13.2 Policies and measurability

An immediate use of our new measure-theory vocabulary is to communicate the relatively simple concept that decisions have to be made without using information from the future. What this section will do is to allow you to talk like a trained stochastic optimizer, but you will also learn a simpler, and perhaps more accurate, way of communicating this simple idea.

As before, let x_t be a decision at time t. The decision x_t, made at time t, depends on the information that has arrived up to time t. The standard mathematical style is to express this dependence by writing the decision as $x_t(\omega)$, where ω represents the sample path as described in section 9.6 and illustrated in table 9.2. It is important to remember that when we use ω, we are specifying the *entire* sample path over the horizon $0, \ldots, T$. This means that we are allowing x_t to "see" not only the entire history, but the entire future!

The probability community has learned how to fix this problem. The following statements all mean the decision x_t depends only on information available up to and including time t:

- "x_t is $\mathcal{F}_t$ measurable." - The fast translation of this statement is that "x_t only uses information that is known at time t." Given our tutorial in the previous subsection, we can provide a little more background. Recall (from above) that $\mathfrak{F}_t$ is a set of sets, where one of the sets in $\mathfrak{F}_t$ will consist of all the sample paths ω that have the same history $h_t(\omega) = (W_1, \ldots, W_t)$ but without regard to the outcomes of $W_{t+1}, \ldots, W_T$. Let $\mathcal{E}_t(h_t)$ be the elementary event that includes all ω where $h_t(\omega) = h_t$ (remember that $\mathfrak{F}_t$ consists of all unions and complements, which means that $\mathfrak{F}_t$ will include events of all ω where $h_{t-1}(\omega) = h_{t-1}$).

  Now, any sample path belonging to $\mathcal{E}_t(h_t)$ should produce the same decision x_t. So, for each elementary event $\mathcal{E}_t(h_t)$ (remember that there is one h_t for each ω) there is a decision, which means we can create a set of decisions that we will call $\mathfrak{X}_t$, where there is a one-to-one correspondence between sets in $\mathfrak{F}_t$ and sets in $\mathfrak{X}_t$.

  Assuming that the sample paths ω are discrete (this would be the case whenever we use a sampled set of ω's), we assume that we have a probability $p(\omega)$ for each ω (probabilists refer to $p^W(\omega)$ as a *measure*). We can find the probability that each set of decisions in $\mathfrak{X}_t$ occur by finding the corresponding set of $\omega's$ in $\mathfrak{F}_t$. So if $\mathcal{E}_t$ is an elementary set in $\mathfrak{F}_t$, we can compute its probability using

  $$P(\mathcal{E}_t) = \sum_{\omega \in \mathcal{E}_t} p^W(\omega).$$

  Then, for each elementary event $\mathcal{E}_t$ there is a single decision $x_t(\mathcal{E}_t)$ which occurs with probability $P(\mathcal{E}_t)$. From this thinking, we can compute the probability of each event in $\mathfrak{X}_t$. So the measure on the sets in $\mathfrak{X}_t$ are computed from the probabilities we already computed from the probabilities in $\mathfrak{F}_t$.

  This is what is meant by saying that a decision x_t is $\mathfrak{F}_t$-measurable.

- "x_t is nonanticipative" - We first encountered "nonanticipativity" in chapter 2 where we introduced "nonanticipativity constraints" in section 2.1.8 when we introduced two-stage stochastic programs (see in particular equation (2.25)). This is really just another way of saying that x_t cannot depend on the actual outcome of $W_{t'}$ for any $t' > t$.

- "x_t is an adapted policy." - This is nothing more than another way of saying that x_t can only depend on what we know up to time t (or that x_t is $\mathfrak{F}_t$-measurable), which in turn means that x_t "adapts" to new information. As we move forward in time, the decision "adapts" to the new information.

- "τ is a stopping time," - For optimal stopping problems, where we are looking to sell an asset at time τ that depends on the price process, we say that "τ is a stopping time," which means that the decision to sell at time $\tau = t$ must be $\mathcal{F}_t$ measurable.

All of these statements require some mathematical sophistication to understand, and they all mean:

$$x_t = X^\pi(S_t) \text{ is a function of the state } S_t.$$

By constructing our policy as depending on the state S_t, we guarantee that the decision does not have access to any information from the future. This follows immediately from

our transition function

$$S_t = S^M(S_{t-1}, x_{t-1}, W_t)$$

which tells us that S_t is only a function of W_t, as well as S_{t-1} and x_{t-1}. By repeating this, we see that S_t is only a function of $S_0, W_1, \ldots, W_t$.

This (much simpler) discussion also brings out that we are not actually interested in the entire history $h_t = (S_0, W_1, \ldots, W_t)$. We really only need the state S_t. For example, in an inventory problem, we only care about how much inventory we have at time t, but if we want to compute the probability of a decision x_t, we need the probability of being in the state S_t, which means we need to know the set of outcomes ω that led us to state S_t. Not surprisingly, this also depends on the prior decisions $x_0, \ldots, x_{t-1}$, which depends on the policy that produced these decisions. Sounds complicated, but we will never actually need to compute the probability of a decision. What we do need is the expected performance of the policy, which we estimate using simulation.

We can conclude from this discussion that you do not need to understand "x_t is $\mathfrak{F}_t$-measurable," beyond understanding that it just means that x_t only has access to information that has arrived on or before time t. All you really need to understand is that x_t depends only on the state S_t, but this means you need to understand what a state variable is. Every theoretician working in stochastic optimization understands "$\mathfrak{F}_t$-measurable," but there are many who do not know what a state variable is.

9.14 LOOKING FORWARD

We are not quite done with modeling. Chapter 10 addresses the rich area of modeling uncertainty which comes in a number of forms. For some applications, it can easily be argued that modeling uncertainty is more important than pursuing optimal policies. However, even books have to limit what they can cover.

After we provide a basic introduction to uncertainty modeling, the rest of the book focuses on designing policies. This material is organized as follows:

Designing policies (chapter 11) - This chapter describes four fundamental (meta) classes of policies, called policy function approximations (PFAs), cost function approximations (CFAs), policies based on value function approximations (VFAs), and direct lookahead policies (DLAs).

Policy function approximations (chapter 12) - The simplest class of policies are policy function approximations, which is where we describe a policy as some sort of analytical function (lookup tables, parametric or nonparametric functions)

Cost function approximations (chapter 13) - Here we find approximations of cost functions which we then minimize (possibly subject to a set of constraints, which we might also modify).

Value function approximations (chapters 14 - 18) - These chapters develop policies based on value functions. Given the richness of this general approach, we present this material in a series of chapters as follows:

Exact dynamic programming (chapter 14) - This is the classical material on dynamic programs with discrete states, discrete actions, and randomness that is simple enough that we can take expectations.

Backward approximate dynamic programming (chapter 15) - This is the first of a series of chapters that present iterative methods for learning approximations of value functions. In this chapter, we introduce a technique we call *backward approximate dynamic programming* since it builds on classical "backward" methods of Markov decision processes presented in chapter 14. The rest of the material on approximate value functions focuses on "forward" methods.

Forward approximate dynamic programming I (chapter 16) - We begin with a presentation of methods for approximating value functions using forward methods. In this chapter, the policy is fixed.

Forward approximate dynamic programming II (chapter 17) - We build on the tools in chapter 16 but now we use our approximate value functions to define our policy.

Forward approximate dynamic programming III (chapter 18) - This chapter focuses on the important special case where the value function is convex in the state variable. This arises in a applications that involve the allocation of resources.

Direct lookahead approximations (chapter 19) - The last class of policies optimizes an approximate lookahead model. We deal with two important problem classes: where decisions are discrete (or discretized), and it is possible to enumerate all actions, and where the decision x is a vector, making it impossible to enumerate all actions.

Multiagent modeling and learning (chapter 20) - We close by addressing the important topic of multiagent modeling, which arises in a wide array of applications, from controlling a fleet of drones or robots, modeling teams of medical technicians or soldiers, or modeling a global supply chain. Multiagent modeling introduces the need for modeling communication, an issue that does not arise in this chapter. We start by modeling basic learning problems as a two-agent system.

9.15 BIBLIOGRAPHIC NOTES

This chapter is a revised version of Chapter 5 from Powell (2011). To our knowledge, this book (and its predecessor in Powell (2011)) are the only books to clearly articulate the five elements of sequential decision problems in this way. However, as we review in Powell (2021) (available on arXiv), our framework closely follows the general style used throughout the optimal control community, with a few minor tweaks, and some major ones. We view the minor tweaks as consisting of:

- We switch from the standard notation of the controls community that uses state x_t and "control" u_t to reflect the substantial community in math programming that uses x for decisions, and we adopt the standard (and more mnemonic) S_t for state (we use a capital letter following the standard style of the applied probability community).

- We use $S^M(s, x, w)$ for the transition function rather than $f(s, x, w)$ for the simple reason that "$f(\cdot)$" is too popular for modeling a wide range of functions. $S^M(\cdot)$ has the mnemonic "state model" or "system model."

- The controls community often writes

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

where w_t is random at time t (see, for example, Bertsekas (2017). This notation is inherited from continuous time models. We use

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

where W_{t+1} is random at time t, but known at $t+1$. This notation allows us to keep to the convention that any variable indexed by t is known at time t.

It is surprisingly common in the controls literature to see people writing the objective function (for stochastic problems) as

$$\min_{u_0,\ldots,u_T} \mathbb{E} \sum_{t=0}^{T} p_t u_t,$$

where (for this example) the prices p_t vary randomly over time (there may be other random elements in the constraints). The problem is that writing $\min_{u_0,\ldots,u_T}$ does not recognize that u_t is a random variable. Mathematically sophisticated authors understand that u_t is random, and can be written $u_t(\omega)$ where ω is a sample path of any random information. It is important to require that "u_t be $\mathcal{F}_t$-measurable," which recognizes that u_t is a function, but it does not provide any indication of how to construct the policy. Often, authors are simply assuming that we will find an optimal policy by solving the Hamilton-Jacobi-Bellman equations, without recognizing that this is often not possible (even approximately).

Our modeling style would write the objective function as

$$\max_{\pi} \mathbb{E} \left\{ \sum_{t=0}^{T} C(S_t, X^\pi(S_t)) | S_0 \right\}$$

where we explicitly search over policies (the switch from min to max is simple preference). Then, we identify four specific classes of policies, which brings transparency to all the approaches that might be used.

Our modeling approach, then, clearly separates the model (which requires searching over policies) from how we solve the model, which we do by designing policies from the four classes.

Section 9.3 - Figure 9.2 which describes the mapping from continuous to discrete time was outlined for me by Erhan Cinlar.

Section 9.4 - The definition of states is amazingly confused in the literature on sequential decision problems. The first recognition of the difference between the physical state and the belief state appears to be in Bellman & Kalaba (1959) which used the term "hyperstate" to refer to the belief state, making the distinction from "physical states" which, even today, are equated by many authors with "state variable."

The control literature has long used state to represent a sufficient statistic (see for example Kirk (2012)), representing the information needed to model the system forward in time. For an introduction to partially observable Markov decision processes, see White (1991). An excellent description of the modeling of Markov decision processes from an AI perspective is given in Boutilier et al. (1999), including a very nice discussion of factored representations of state variables. See also Guestrin et al. (2003) for an application of the concept of factored state spaces to a Markov decision process.

The definition of a state variable here refines the definition introduced in Powell (2011).

Section 9.5 - Our notation for decisions represents an effort to bring together the fields of dynamic programming and math programming. We believe this notation was first used in Powell et al. (2001). For a classical treatment of decisions from the perspective of Markov decision processes, see Puterman (2005). For examples of decisions from the perspective of the optimal control community, see Kirk (2012) and Lewis & Vrabie (2012). For examples of treatments of dynamic programming in economics, see Stokey & R. E. Lucas (1989) and Chow (1997).

Section 9.6 - Our representation of information follows classical styles in the probability literature (see, for example, Chung (1974)). Considerable attention has been given to the topic of supervisory control (see, for example Werbos (1992)).

Section 9.7 - The concept of a "transition function" (which has been given a number of different names) is absolutely standard in the controls community, and yet surprisingly absent throughout the other communities (and in particular Markov decision processes, which seems to insist on using one-step transition matrices). See any of the books on optimal control (Kirk (2012), Stengel (1986), Sontag (1998), Sethi (2019), and Lewis & Vrabie (2012)). Bertsekas (2017) opens his book by stating the transition function, but then switches to using transition matrices (or kernels for continuous states), which require taking the expectation of the transition function.

Section 9.11 - Our identification of the four classes of objectives (final reward and cumulative reward, state-independent problems and state-dependent problems) was first presented in Powell (2019), although the material in section 9.12 is new.

Section 9.9 - The energy storage example is taken from Powell (2021) (also available on arXiv).

EXERCISES

Review questions

9.1 What is the difference between the *history* of a process, and the state of a process?

9.2 What is meant by a "martingale model of forecast evolution"?

9.3 What are the five components of a sequential decision problem?

9.4 What are the three types of state variables? Give an example of each.

9.5 What may the exogenous information W_{t+1} depend on?

9.6 Assuming the state S_t is discrete, how do you compute the one-step probability transition matrix from the transition function, assuming you know the probability distribution of W_{t+1}?

9.7 Write out, and explain, the objective functions for the following four cases:

1) State-independent problems, final reward.

2) State-independent problems, cumulative reward.

3) State-dependent problems, cumulative reward.

4) State-dependent problems, final reward.

Modeling questions

9.8 A traveler needs to traverse the graph shown in figure 9.7 from node 1 to node 11 where the goal is to find the path that minimizes the sum of the costs over the path. To solve this problem we are going to use the deterministic version of Bellman's optimality equation that states

$$V(s) = \min_{a \in \mathcal{A}_s} \big(c(s,a) + V(s'(s,a))\big) \tag{9.55}$$

where $s'(s,a)$ is the state we transition to when we are in state s and take action $a \in \mathcal{A}_s$. The set $\mathcal{A}_s$ is the set of actions (in this case, traversing over a link) available when we are in state s.

To solve this problem, answer the following questions:

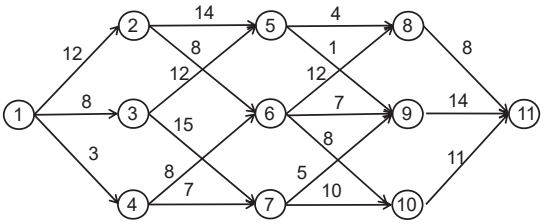


Figure 9.7 Deterministic shortest path problem.

a) Describe an appropriate state variable for this problem (with notation).

b) If the traveler is at node 6 by following the path 1-2-6, what is her state?

c) Find the path that minimizes the sum of the costs on the links traversed by the traveler. Using Bellman's equation (14.2), work backwards from node 11 and find the best path from each node to node 11, ultimately finding the best path from node 1 to node 11. Show your solution by drawing the graph with the links that fall on an optimal path from some node to node 11 drawn in bold.

9.9 A traveler needs to traverse the graph shown in figure 9.8 from node 1 to node 11, where the goal is to find the path that minimizes the *largest* cost of all the links on the path. To solve this problem, answer the following questions:

a) Describe an appropriate state variable for this problem (with notation).

b) If the traveler is at node 6 by following the path 1-2-6, what is her state?

c) Using Bellman's equation (14.2), find the path (or paths) that minimizes the largest costs on the links traversed by the traveler. For each decision point (the nodes in the graph), give the value of the state variable corresponding to the optimal path to that

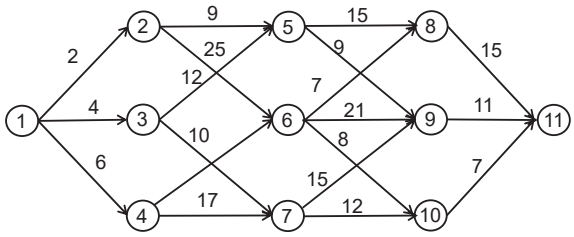


Figure 9.8 A path problem minimizing the largest cost of all the links on a path.

decision point, and the value of being in that state (that is, the cost if we start in that state and then follow the optimal solution).

9.10 Repeat exercise 9.9, but this time minimize the *second largest* arc cost on a path.

9.11 A traveler needs to traverse the graph shown in figure 9.9 from node 1 to node 11, where the goal is to find the path that minimizes the *second largest* link cost along the path. To solve this problem, answer the following questions:

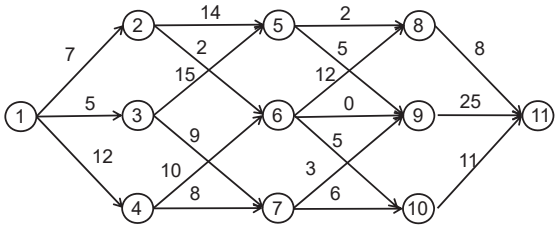


Figure 9.9 A path problem minimizing the product of the costs.

a) Describe an appropriate state variable for this problem (with notation).

b) If the traveler is at node 6 by following the path 1-2-6, what is her state?

c) If the traveler is at node 10 by following the path 1-2-6-10, what is her state?

d) Using Bellman's equation (14.2), find the path (or paths) that minimizes the product of the costs on the links traversed by the traveler. For each decision point (the nodes in the graph), give the value of the state variable corresponding to the optimal path to that decision point, and the value of being in that state (that is, the cost if we start in that state and then follow the optimal solution).

9.12 Consider our basic newsvendor problem

$$\max_x \mathbb{E}_D F(x, D) = \mathbb{E}_D\big(p \min\{x, D\} - cx\big). \tag{9.56}$$

Show how the following variations of this problem can be modeled using the universal modeling framework:

a) The final reward formulation of the basic newsvendor problem.

b) The cumulative reward formulation of the basic newsvendor problem.

c) The asymptotic formulation of the newsvendor problem. What are the differences between the asymptotic formulation and the final reward formulation?

9.13 Now consider a dynamic version of our newsvendor problem where a decision x_t is made at time t by solving

$$\max_x \mathbb{E}_D F(x, D) = \mathbb{E}_D\big(p_t \min\{x, D_{t+1}\} - cx\big). \tag{9.57}$$

Assume that the price p_t is independent of prior history.

a) Model the cumulative reward version of the newsvendor problem in (9.57).

b) How does your model change if we are instead solving

$$\max_x \mathbb{E}_D F(x, D) = \mathbb{E}_D\big(p_{t+1} \min\{x, D_{t+1}\} - cx\big). \tag{9.58}$$

where we continue to assume that the price, which is now p_{t+1}, is independent of prior history.

c) How does your model of (9.58) change if

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \varepsilon_{t+1}, \tag{9.59}$$

where ε_{t+1} is a zero-mean noise term, independent of the state of the system.

9.14 We continue the newsvendor problem in exercise 9.13, but now assume that (θ_0, θ_1) in equation (9.59) are unknown. At time t, we have estimates $\bar{\theta}_t = (\bar{\theta}_{t0}, \bar{\theta}_{t1}$. Assume the true θ is now a random variable that follows a multivariate normal distribution with mean $\mathbb{E}_t\theta = \bar{\theta}_t$ which we initialize to

$$\bar{\theta}_0 = \begin{pmatrix} 20 \\ 40 \end{pmatrix},$$

and covariance matrix Σ_t^θ which we initialize to

$$\begin{aligned} \Sigma_0^\theta &= \begin{pmatrix} \sigma_{00}^2 & \sigma_{01}^2 \\ \sigma_{10}^2 & \sigma_{11}^2 \end{pmatrix}. \\ &= \begin{pmatrix} 36 & 16 \\ 16 & 25 \end{pmatrix}. \end{aligned}$$

Drawing on the updating equations in section 3.4.2, give a full model of this problem using a cumulative reward objective function (that is, give the state, decision and exogenous information variables, transition function and objective unction).

9.15 Below is a series of variants of our familiar newsvendor (or inventory) problem. In each, describe the pre- and post-decision states, decision and exogenous information in the form:

$$(S_0, x_0, S_0^x, W_1, S_1, x_1, S_1^x, W_2, \ldots)$$

Specify S_t, S_t^x, x_t and W_t in terms of the variables of the problem.

a) The basic newsvendor problem where we wish to find x that solves

$$\max_x \mathbb{E}\{p \min(x, \hat{D}) - cx\} \tag{9.60}$$

where the distribution of $\hat{D}$ is unknown.

b) The same as (a), but now we are given a price p_t at time t and asked to solve (9.60) using this information. Note that p_t is unrelated to any prior history or decisions.

c) Repeat (b), but now $p_{t+1} = p_t + \hat{p}_{t+1}$.

d) Repeat (c), but now leftover inventory is held to the next time period.

e) Of the problems above, which (if any) are *not* dynamic programs? Explain.

f) Of the problems above, which would be classified as solving state-dependent vs. state-independent functions.

9.16 In this exercise you are going to model an energy storage problem, which is a problem class that arises in many settings (how much cash to keep on hand, how much inventory on a store shelf, how many units of blood to hold, how many milligrams of a drug to keep in a pharmacy, ...). We will begin by describing the problem in English with a smattering of notation. Your job will be to develop it into a formal dynamic model.

Our problem is to decide how much energy to purchase from the electric power grid at a price p_t. Let x_t^{gs} be the amount of power we buy (if $x^{gs} > 0$) or sell (if $x^{gs} < 0$). We then have to decide how much energy to move from storage to meet the demand D_t in a commercial building, where $x_t^{sb} \geq 0$ is the amount we move to the building to meet the demand D_t. Unsatisfied demand is penalized at a price c per unit of energy.

Assume that prices evolve according to a time-series model given by

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}, \tag{9.61}$$

where ε_{t+1} is a random variable with mean 0 that is independent of the price process. We do not know the coefficients θ_i for $i = 0, 1, 2$, so instead we use estimates $\bar{\theta}_{ti}$. As we observe p_{t+1}, we can update the vector $\bar{\theta}_t$ using the recursive formulas for updating linear models as described in chapter 3, section 3.8 (you will need to review this section to answer parts of this question).

Every time period we are given a forecast $f_{tt'}^D$ of the demand $D_{t'}$ at time t' in the future, where $t' = t, t+1, t+H$. We can think of $f_{tt}^D = D_t$ as the actual demand. We can also think of the forecasts $f_{t+1,t'}^D$ as the "new information" or define a "change in the forecast" $\hat{f}_{t+1,t'}^D$ in which case we would write

$$f_{t+1,t'}^D = f_{tt'}^D + \hat{f}_{t+1,t'}^D.$$

a) What are the elements of the state variable S_t (we suggest filling in the other elements of the model to help identify the information needed in S_t). Define both the pre- and post-decision states.

b) What are the elements of the decision variable x_t? What are the constraints (these are the equations that describe the limits on the decisions). Finally introduce a function

$X^\pi(S_t)$ which will be our policy for making decisions to be designed later (but we need it in the objective function below).

c) What are the elements of the exogenous information variable W_{t+1} that become known at time $t+1$ but which were not known at time t.

d) Write out the transition function $S_{t+1} = S^M(S_t, x_t, W_{t+1})$, which is the equations that describe how each element of the state variable S_t evolves over time. There needs to be one equation for each state variable.

e) Write out the objective function by writing:

The contribution function $C(S_t, x_t)$.

The objective function where you maximize expected profits over some general set of policies (to be defined later - not in this exercise).

9.17 Patients arrive at a doctor's office, each of whom are described by a vector of attribute $a = (a_1, a_2, \ldots, a_K)$ where a might describe age, gender, height, weight, whether the patient smokes, and so on. Let a^n be the attribute vector describing the n^{th} patient. For each patient, the doctor makes a decision x^n (surgery, drug regimens, rehabilitation), and then observes an outcome y^n for patient n. From y^n, we obtain an updated estimate θ^n for the parameters of a nonlinear model $f(x|\theta)$ that helps us to predict y for other patients.

a) Give the five elements of this decision problem. Be sure to model the state after a patient arrives (this would be the pre-decision state), S^n, after a decision is made (this would be the post-decision state), $S^{x,n}$ and after the outcome of a decision becomes known, $S^{y,n}$.

b) The value of being in a state S^n can be computed using Bellman's equation

$$V^n(S^n) = \max_{x \in \mathcal{X}} \big((C(S^n, x) + E_W\{V^{n+1}(S^{n+1})|S^n, x\} \big). \quad (9.62)$$

Define the value of being in the state i) after a patient arrives, ii) after a decision is made, and iii) before a patient arrives. Call these $V(S)$, $V^x(S^x)$, and $V^y(S^y)$. Write $V(S^n)$ as a function of $V^x(S^{x,n})$ and write $V^x(S^{x,n})$ as a function of $V^y(S^{y,n})$.

9.18 Consider the problem of controlling the amount of cash a mutual fund keeps on hand. Let R_t be the cash on hand at time t. Let $\hat{R}_{t+1}$ be the net deposits (if $\hat{R}_{t+1} > 0$) or withdrawals (if $\hat{R}_{t+1} < 0$), where we assume that $\hat{R}_{t+1}$ is independent of $\hat{R}_t$. Let M_t be the stock market index at time t, where the evolution of the stock market is given by $M_{t+1} = M_t + \hat{M}_{t+1}$ where $\hat{M}_{t+1}$ is independent of M_t. Let x_t be the amount of money moved from the stock market into cash ($x_t > 0$) or from cash into the stock market ($x_t < 0$).

a) Give a complete model of the problem, including both pre-decision and post-decision state variables.

b) Suggest a simple parametric policy function approximation, and give the objective function as an online learning problem.

9.19 A college student must plan what courses she takes over each of eight semesters. To graduate, she needs 34 total courses, while taking no more than five and no less than

three courses in any semester. She also needs two language courses, one science course, eight departmental courses in her major and two math courses.

(a) Formulate the state variable for this problem in the most compact way possible.

(b) Give the transition function for our college student assuming that she successfully passes any course she takes. You will need to introduce variables representing her decisions.

(c) Give the transition function for our college student, but now allow for the random outcome that she may not pass every course.

9.20 A broker is working in thinly traded stocks. He must make sure that he does not buy or sell in quantities that would move the price and he feels that if he works in quantities that are no more than 10 percent of the average sales volume, he should be safe. He tracks the average sales volume of a particular stock over time. Let $\hat{v}_t$ be the sales volume on day t, and assume that he estimates the average demand f_t using $f_t = (1-\alpha)f_{t-1} + \alpha\hat{v}_t$. He then uses f_t as his estimate of the sales volume for the next day. Assuming he started tracking demands on day $t = 1$, what information would constitute his state variable?

9.21 How would your previous answer change if our broker used a 10-day moving average to estimate his demand? That is, he would use $f_t = 0.10\sum_{i=1}^{10}\hat{v}_{t-i+1}$ as his estimate of the demand.

9.22 The pharmaceutical industry spends millions managing a sales force to push the industry's latest and greatest drugs. Assume one of these salesmen must move between a set $\mathcal{I}$ of customers in his district. He decides which customer to visit next only after he completes a visit. For this exercise, assume that his decision does not depend on his prior history of visits (that is, he may return to a customer he has visited previously). Let S_n be his state immediately after completing his n^{th} visit that day.

(a) Assume that it takes exactly one time period to get from any customer to any other customer. Write out the definition of a state variable, and argue that his state is only his current location.

(b) Now assume that τ_{ij} is the (deterministic and integer) time required to move from location i to location j. What is the state of our salesman at any time t? Be sure to consider both the possibility that he is at a location (having just finished with a customer) or between locations.

(c) Finally assume that the travel time τ_{ij} follows a discrete uniform distribution between a_{ij} and b_{ij} (where a_{ij} and b_{ij} are integers)?

9.23 Consider a simple asset acquisition problem where x_t is the quantity purchased at the end of time period t to be used during time interval $t+1$. Let D_t be the demand for the assets during time interval t. Let R_t be the pre-decision state variable (the amount on hand before you have ordered x_t) and R_t^x be the post-decision state variable.

(a) Write the transition function so that R_{t+1} is a function of R_t, x_t, and D_{t+1}.

(b) Write the transition function so that R_t^x is a function of R_{t-1}^x, D_t, and x_t.

(c) Write R_t^x as a function of R_t, and write R_{t+1} as a function of R_t^x.

9.24 As a buyer for an orange juice products company, you are responsible for buying futures for frozen concentrate. Let $x_{tt'}$ be the number of futures you purchase in year t that can be exercised during year t'.

(a) What is your state variable in year t?

(b) Write out the transition function.

9.25 A classical inventory problem works as follows. Assume that our state variable R_t is the amount of product on hand at the end of time period t and that D_t is a random variable giving the demand during time interval $(t-1, t)$ with distribution $p_d = P(D_t = d)$. The demand in time interval t must be satisfied with the product on hand at the beginning of the period. We can then order a quantity x_t at the end of period t that can be used to replenish the inventory in period $t+1$. Give the transition function that relates R_{t+1} to R_t.

9.26 Many problems involve the movement of resources over networks. The definition of the state of a single resource, however, can be complicated by different assumptions for the probability distribution for the time required to traverse a link. For each example below, give the state of the resource:

(a) You have a deterministic, static network, and you want to find the shortest path from an origin node q to a destination node r. There is a known cost c_{ij} for traversing each link (i, j).

(b) Next assume that the cost c_{ij} is a random variable with an unknown distribution. Each time you traverse a link (i, j), you observe the cost $\hat{c}_{ij}$, which allows you to update your estimate $\bar{c}_{ij}$ of the mean of c_{ij}.

(c) Finally assume that when the traveler arrives at node i he sees $\hat{c}_{ij}$ for each link (i, j) out of node i.

(d) A taxicab is moving people in a set of cities $\mathcal{C}$. After dropping a passenger off at city i, the dispatcher may have to decide to reposition the cab from i to j, $(i, j) \in \mathcal{C}$. The travel time from i to j is τ_{ij}, which is a random variable with a discrete uniform distribution (that is, the probability that $\tau_{ij} = t$ is $1/T$, for $t = 1, 2, \ldots, T$). Assume that the travel time is known before the trip starts.

(e) Same as (d), but now the travel times are random with a geometric distribution (that is, the probability that $\tau_{ij} = t$ is $(1-\theta)\theta^{t-1}$, for $t = 1, 2, 3, \ldots$).

9.27 As the purchasing manager for a major citrus juice company, you have the responsibility of maintaining sufficient reserves of oranges for sale or conversion to orange juice products. Let x_{ti} be the amount of oranges that you decide to purchase from supplier i in week t to be used in week $t+1$. Each week, you can purchase up to $\hat{q}_{ti}$ oranges (that is, $x_{ti} \leq \hat{q}_{ti}$) at a price $\hat{p}_{ti}$ from supplier $i \in \mathcal{I}$, where the price/quantity pairs $(\hat{p}_{ti}, \hat{q}_{ti})_{i \in \mathcal{I}}$ fluctuate from week to week. Let s_0 be your total initial inventory of oranges, and let D_t be the number of oranges that the company needs for production during week t (this is our demand). If we are unable to meet demand, the company must purchase additional oranges on the spot market at a spot price $\hat{p}_{ti}^{spot}$.

(a) What is the exogenous stochastic process for this system?

(b) What are the decisions you can make to influence the system?

(c) What would be the state variable for your problem?

(d) Write out the transition equations.

(e) What is the one-period contribution function?

(f) Propose a reasonable structure for a decision rule for this problem, and call it X^π. Your decision rule should be in the form of a function that determines how much to purchase in period t.

(g) Carefully and precisely, write out the objective function for this problem in terms of the exogenous stochastic process. Clearly identify what you are optimizing over.

(h) For your decision rule, what do we mean by the space of policies?

9.28 Customers call in to a service center according to a (nonstationary) Poisson process. Let $\mathcal{E}$ be the set of events representing phone calls, where $t_e, e \in \mathcal{E}$ is the time that the call is made. Each customer makes a request that will require time τ_e to complete and will pay a reward r_e to the service center. The calls are initially handled by a receptionist who determines τ_e and r_e. The service center does not have to handle all calls and obviously favors calls with a high ratio of reward per time unit required (r_e/τ_e). For this reason, the company adopts a policy that the call will be refused if $(r_e/\tau_e) < \gamma$. If the call is accepted, it is placed in a queue to wait for one of the available service representatives. Assume that the probability law driving the process is known, where we would like to find the right value of γ.

(a) This process is driven by an underlying exogenous stochastic process with element $\omega \in \Omega$. What is an instance of ω?

(b) What are the decision epochs?

(c) What is the state variable for this system? What is the transition function?

(d) What is the action space for this system?

(e) Give the one-period reward function.

(f) Give a full statement of the objective function that defines the Markov decision process. Clearly define the probability space over which the expectation is defined, and what you are optimizing over.

9.29 A major oil company is looking to build up its storage tank reserves, anticipating a surge in prices. It can acquire 20 million barrels of oil, and it would like to purchase this quantity over the next 10 weeks (starting in week 1). At the beginning of the week, the company contacts its usual sources, and each source $j \in \mathcal{J}$ is willing to provide $\hat{q}_{tj}$ million barrels at a price $\hat{p}_{tj}$. The price/quantity pairs $(\hat{p}_{tj}, \hat{q}_{tj})$ fluctuate from week to week. The company would like to purchase (in discrete units of millions of barrels) x_{tj} million barrels (where x_{tj} is discrete) from source j in week $t \in \{1, 2, \ldots, 10\}$. Your goal is to acquire 20 million barrels while spending the least amount possible.

(a) What is the exogenous stochastic process for this system?

(b) What would be the state variable for your problem? Give an equation(s) for the system dynamics.

(c) Propose a structure for a decision rule for this problem and call it X^π.

(d) For your decision rule, what do we mean by the space of policies? Give examples of two different decision rules.

(e) Write out the objective function for this problem using an expectation over the exogenous stochastic process.

(f) You are given a budget of \$300 million to purchase the oil, but you absolutely must end up with 20 million barrels at the end of the 10 weeks. If you exceed the initial budget of \$300 million, you may get additional funds, but each additional \$1 million will cost you \$1.5 million. How does this affect your formulation of the problem?

9.30 You own a mutual fund where at the end of each week t you must decide whether to sell the asset or hold it for an additional week. Let $\hat{r}_t$ be the one-week return (e.g. $\hat{r}_t = 1.05$ means the asset gained five percent in the previous week), and let p_t be the price of the asset if you were to sell it in week t (so $p_{t+1} = p_t\hat{r}_{t+1}$). We assume that the returns $\hat{r}_t$ are independent and identically distributed. You are investing this asset for eventual use in your college education, which will occur in 100 periods. If you sell the asset at the end of time period t, then it will earn a money market rate q for each time period until time period 100, at which point you need the cash to pay for college.

(a) What is the state space for our problem?

(b) What is the action space?

(c) What is the exogenous stochastic process that drives this system? Give a five time period example. What is the history of this process at time t?

(d) You adopt a policy that you will sell if the asset falls below a price $\bar{p}$ (which we are requiring to be independent of time). Given this policy, write out the objective function for the problem. Clearly identify exactly what you are optimizing over.

Theory questions

9.31 Assume that we have N discrete resources to manage, where R_a is the number of resources of type $a \in \mathcal{A}$ and $N = \sum_{a\in\mathcal{A}} R_a$. Let $\mathcal{R}$ be the set of possible values of the vector R. Show that

$$|\mathcal{R}| = \left(\begin{array}{c} N + |\mathcal{A}| - 1 \\ |\mathcal{A}| - 1 \end{array} \right),$$

where

$$\left(\begin{array}{c} X \\ Y \end{array} \right) = \frac{X!}{Y!(X-Y)!}$$

is the number of combinations of X items taken Y at a time.

Diary problem

The diary problem is a single problem you chose (see chapter 1 for guidelines). Answer the following for your diary problem.

9.32 Now you are finally going to model your diary problem, in its full detail (but you will not attempt to design a policy).

a) Define each of the elements of the state variable. Note that this is an iterative process; you generally need to define the state variable as you identify the information you need at time t to model the system from time t onward. Do you have a belief state? If not, try to introduce one. All you need is some parameter that you can model as being unknown, but which you can estimate as data arrives to the system. The most interesting problems are where your decisions influence what you observe.

b) What are the decisions? Describe in words, and then introduce notation for each decision. Now describe the constraints or the set of allowable decisions at time t. Add any information that you need at time t (that may change as we step forward in time) to the state variable. Introduce notation for the policy, although we will design the policy after we complete the model. The policy may introduce additional information that will have to be added to the state variable, but we will handle this after we start to design the policy.

c) What is the exogenous information that arrives after you make the decision? (Note that you may have a deterministic problem, which means you do not have any exogenous information.) If your exogenous information depends on what you know at time t, then this information must be in the state variable.

d) Define the transition function, which describes how each state variable evolves over time (not that we may not be done with the state variable). The level of detail here will depend on the complexity of your problem. Feel free to use both model-based transitions (where the equation governing the transition is known) and model-free transitions (where you simply observe the updated value of the variable).

e) Write out the one-period contribution function, which may introduce additional information that you will need to add to the state variable (with corresponding additions to the transition function). Now write out the value of a policy, and write the objective of maximizing over policies (or classes of policies).

Bibliography

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

Bellman, R. E. & Kalaba, R. (1959), 'On adaptive control processes', *OIRE Trans.* **4**, 1–9.

Bertsekas, D. P. (2017), *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, 4 edn, Athena Scientific, Belmont, MA.

Boutilier, C., Dean, T. & Hanks, S. (1999), 'Decision-Theoretic Planning: Structural Assumptions and Computational Leverage', *Access* pp. 1–94.

Chow, G. (1997), *Dynamic Economics*, Oxford University Press, New York.

Chung, K. L. (1974), *A Course in Probability Theory*, Academic Press, New York.

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

Guestrin, C., Koller, D. & Parr, R. (2003), 'Efficient Solution Algorithms for Factored MDPs', *Journal of Artificial Intelligence Research* **19**, 399–468.

Kirk, D. E. (2012), *Optimal Control Theory: An introduction*, Dover, New York.

Lewis, F. L. & Vrabie, D. (2012), *Design Optimal Adaptive Controllers*, 3 edn, John Wiley & Sons, Hoboken, NJ.

Pollard, D. (2002), *A User's Guide to Measure Theoretic Probability*, Cambridge University Press, Cambridge.

Powell, W. B. (2011), *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2 edn, John Wiley & Sons.

Powell, W. B. (2019), 'A unified framework for stochastic optimization', *European Journal of Operational Research* **275**(3), 795–821.

Powell, W. B. (2021), 'From reinforcement learning to optimal control: A unified framework for sequential decisions', *Handbook on Reinforcement Learning and Optimal Control, Studies in Systems, Decision and Control* pp. 29–74.

Powell, W. B., Simao, H. P. & Shapiro, J. A. (2001), A representational paradigm for dynamic resource transformation problems, *in* F. C. Coullard & J. Owens, H., eds, 'Annals of Operations Research', J. C. Baltzer AG, pp. 231–279.

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

Sethi, S. P. (2019), *Optimal Control Theory: Applications to Management Science and Economics*, 3 edn, Springer-Verlag, Boston.

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

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

Stokey, N. L. & R. E. Lucas, J. (1989), *Recursive Methods in Dynamic Economics*, Harvard University Press, Cambridge, MA.

Werbos, P. J. (1992), Neurocontrol and Supervised Learning: an Overview and Evaluation, *in* D. A. White & D. A. Sofge, eds, 'Handbook of Intelligent Control', Von Nostrand Reinhold, New York, pp. 65–86.

White, C. C. (1991), 'A survey of solution techniques for the partially observable Markov decision process', *Annals of operations research* **32**, 215–230.