CHAPTER 5

MODELING DYNAMIC PROGRAMS

Perhaps one of the most important skills to develop in approximate dynamic programming is the ability to write down a model of the problem. Everyone who wants to solve a linear program learns to write out

$$\min_x c^T x$$

subject to

$$\begin{aligned} Ax &= b, \\ x &\geq 0. \end{aligned}$$

This standard modeling framework allows people around the world to express their problem in a standard format.

Stochastic, dynamic problems are much richer than a linear program, and require the ability to model the flow of information and complex system dynamics. Just the same, there is a standard framework for modeling dynamic programs. We provided a taste of this framework in chapter 2, but that chapter only hinted at the richness of the problem class.

In chapters 2, 3 and 4, we used fairly standard notation, and have avoided discussing some important subtleties that arise in the modeling of stochastic, dynamic systems. We intentionally overlooked trying to define a state variable, which we have viewed as simply S_t, where the set of states was given by the indexed set $\mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}$. We have avoided discussions of how to properly model time or more complex information processes. This style has facilitated introducing some basic ideas in dynamic programming, but would severely limit our ability to apply these methods to real problems.

The goal of this chapter is to describe a standard modeling framework for dynamic programs, providing a vocabulary that will allow us to take on a much wider set of applications. 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 making the model completely intractable as problems become more complex. Complex problems require considerable discipline in notation because they combine the details of the original physical problem with the challenge of modeling sequential information and decision processes. The modeling of time can be particularly subtle. In addition to a desire to model problems accurately, we also need to be able to understand and exploit the structure of the problem, which can become lost in a sea of complex notation.

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. For example, it is common in dynamic programming to refer to actions using "a" (where a is discrete); in control theory a decision (control) is "u" (which may be continuous). For high-dimensional problems, it is essential to draw on the field of mathematical programming, where decisions are typically written as "x" and resource constraints are written in the standard form $Ax = b$. In this text, many of our problems involve managing resources where we are trying to maximize or minimize an objective subject to constraints. For this reason, we adopt, as much as possible, the notation of math programming to help us bridge the fields of math programming and dynamic programming.

Sections 5.1 to 5.3 provide some foundational material. Section 5.1 begins by describing some basic guidelines for notational style. Section 5.2 addresses the critical question of modeling time, and section 5.3 provides notation for modeling resources that we will use throughout the remainder of the volume.

The general framework for modeling a dynamic program is covered in sections 5.4 to 5.8. There are five elements to a dynamic program, consisting of the following:

1) State variables - These describe what we need to know at a point in time (section 5.4).

2) Decision variables - These are the variables we control. Choosing these variables ("making decisions") represents the central challenge of dynamic programming (section 5.5).

3) Exogenous information processes - These variables describe information that arrives to us exogenously, representing the sources of randomness (section 5.6).

4) Transition function - This is the function that describes how the state evolves from one point in time to another (section 5.7).

5) Objective function - We are either trying to maximize a contribution function (profits, revenues, rewards, utility) or minimize a cost function. This function describes how well we are doing at a point in time (section 5.8).

This chapter describes modeling in considerable depth, and as a result it is quite long. A number of sections are marked with a '*', indicating that these can be skipped on a first read. There is a single section marked with a '**' which, as with all sections marked this way, is material designed for readers with more advanced training in probability and stochastic processes.

5.1 NOTATIONAL STYLE

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 "holding cost."

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.

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. If we write x_{ti}, this would be the vector of flows out of i at time t to all destinations j. Time, when present, is always the innermost index.

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 $x = (x_1, x_2, \ldots, x_t, \ldots, x_T)$ 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).

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 - We place iteration counters in the superscript, and we primarily use n as our iteration counter. So, x^n is our activity at iteration n. 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. We make one notable exception to our policy of indexing iterations in the superscript. In approximate dynamic programming, we make wide use of a parameter known as a stepsize α where $0 \leq \alpha \leq 1$. We often make the stepsize vary with the iterations. However, writing α^n looks too much like raising the stepsize to the power of n. Instead, we write α_n to indicate the stepsize in iteration n. This is our only exception to this rule.

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.

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

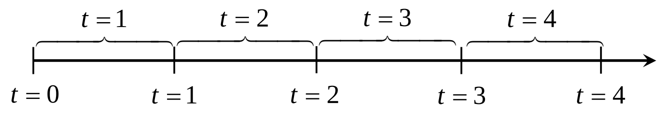
Of course, there are exceptions to every rule. It is extremely common in the transportation literature to model the flow of a type of resource (called a commodity and indexed by k) from i to j using x^k_{ij}. Following our convention, this should be written x_{kij}. Authors need to strike a balance between a standard notational style and existing conventions.

5.2 MODELING TIME

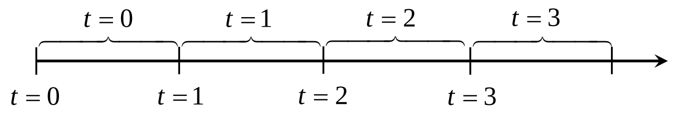
A survey of the literature reveals different styles toward modeling time. When using discrete time, some authors start at 1 while others start at zero. When solving finite horizon problems, it is popular to index time by the number of time periods remaining, rather than elapsed time. Some authors index a variable, say S_t, as being a function of information up through $t-1$, while others assume it includes information up through time t. t may be used to represent when a physical event actually happens, or when we first know about a physical event.

The confusion over modeling time arises in large part because there are two processes that we have to capture: the flow of information, and the flow of physical and financial resources. There are many applications of dynamic programming to deterministic problems where the flow of information does not exist (everything is known in advance). Similarly, there are many models where the arrival of the information about a physical resource, and when the information takes effect in the physical system, are the same. For example, the time at which a customer physically arrives to a queue is often modeled as being the same as when the information about the customer first arrives. Similarly, we often assume that we can sell a resource at a market price as soon as the price becomes known.

There is a rich collection of problems where the information process and physical process are different. 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 (see section 2.2.7 for an illustration).



5.1a: Information processes



5.1b: Physical processes

Figure 5.1 Relationship between discrete and continuous time for information processes (5.1a) and physical processes (5.1b).

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, and many of the physical processes being modeled, are best viewed in continuous time. A common error is to assume that when you model a dynamic program in discrete time then all events (information events and physical events) are also occurring in discrete time (in some applications, this is the case). Throughout this volume, decisions are made in discrete time, while all other activities occur in continuous time.

The relationship of our discrete time approximation to the real flow of information and physical resources is depicted in figure 5.1. 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 5.1a). 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

$$f_t = (1-\alpha)f_{t-1} + \alpha\hat{D}_t. \tag{5.1}$$

We refer to this form as the *informational representation*. Note that the forecast f_t is written as a function of the information that became available during time interval t.

When we are modeling a physical process, it is more natural to adopt a different convention (illustrated in figure 5.1b): 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

$$R_{t+1} = R_t + x_t - \hat{D}_t. \tag{5.2}$$

We refer to this form as the *actionable 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. This equation makes perfect sense when we interpret time t to represent when a quantity can be used. For example, many authors would write our forecasting equation (5.1) as

$$f_{t+1} = (1-\alpha) f_t + \alpha \hat{D}_t. \tag{5.3}$$

This equation is correct if we interpret f_t as the forecast of the demand that will happen in time interval t.

A review of the literature quickly reveals that both modeling conventions are widely used. It is important to be aware of the two conventions and how to interpret them. We handle the modeling of informational and physical processes by using two time indices, a form that we refer to as the "(t, t')" notation. For example,

$\hat{D}_{tt'}$	=	The demands that first become known during time interval t to be served during time interval t'.
$f_{tt'}$	=	The forecast for activities during time interval t' made using the information available up through time t.
$R_{tt'}$	=	The resources on hand at time t that cannot be used until time t'.
$x_{tt'}$	=	The decision to purchase futures at time t to be exercised during time interval t'.

For each variable, t indexes the information content (literally, when the variable is measured or computed), while t' represents the time at which the activity takes place. Each of these variables can be written as vectors, such as

$$\begin{aligned}
\hat{D}_t &= (\hat{D}_{tt'})_{t' \geq t}, \\
f_t &= (f_{tt'})_{t' \geq t}, \\
x_t &= (x_{tt'})_{t' \geq t}, \\
R_t &= (R_{tt'})_{t' \geq t}.
\end{aligned}$$

Note that these vectors are now written in terms of the information content. For stochastic problems, this style is the easiest and most natural. If we were modeling a deterministic problem, we would drop the first index "t" and model the entire problem in terms of the second index "t'."

Each one of these quantities is computed at the end of time interval t (that is, with the information up through time interval t) and represents a quantity that can be used at time t' in the future. We could adopt the convention that the first time index uses the indexing system illustrated in figure 5.1a, while the second time index uses the system in figure 5.1b. While this convention would allow us to easily move from a natural deterministic model to a natural stochastic model, we suspect most people will struggle with an indexing system where time interval t in the information process refers to time interval $t-1$ in the physical process. Instead, we adopt the convention to model information in the most natural way, and live with the fact that product arriving at time t can only be used during time interval $t+1$.

Using this convention it is instructive to interpret the special case where $t = t'$. $\hat{D}_{tt}$ is simply demands that arrive during time interval t, where we first learn of them when they arrive. f_{tt} makes no sense, because we would never forecast activities during time interval t after we have this information. R_{tt} represents resources that we know about during time interval t and which can be used during time interval t. Finally, x_{tt} is a decision to purchase resources to be used during time interval t given the information that arrived during time interval t. In financial circles, this is referred to as purchasing on the spot market.

The most difficult notational decision arises when first starting to work on a problem. It is natural at this stage to simplify the problem (often, the problem *appears* simple) and then choose notation that seems simple and natural. If the problem is deterministic and you are quite sure that you will never solve a stochastic version of the problem, then the actionable representation (figure 5.1b) and equation (5.2) is going to be the most natural. Otherwise, it is best to choose the informational format. If you do not have to deal with lagged information processes (e.g., ordering at time t to be used at some time t' in the future) you should be able to get by with a single time index, but you need to remember that x_t may mean purchasing product to be used during time interval $t+1$.

Care has to be used when taking expectations of functions. Consider what happens when we want to know the expected costs to satisfy customer demands $\hat{D}_t$ that arose during time interval t given the decision x_{t-1} we made at time $t-1$. We would have to compute $\mathbb{E}\{C_t(x_{t-1}, \hat{D}_t)\}$, where the expectation is over the random variable $\hat{D}_t$. The function that results from taking the expectation is now a function of information up through time $t-1$. Thus, we might use the notation

$$\bar{C}_{t-1}(x_{t-1}) = \mathbb{E}\{C_t(x_{t-1}, \hat{D}_t)\}.$$

This can take a little getting used to. The costs are incurred during time interval t, but now we are indexing the function with time $t-1$. The problem is that if we use a single time index, we are not capturing when the activity is actually happening. An alternative is to switch to a double time index, as in

$$\bar{C}_{t-1,t}(x_{t-1}) = \mathbb{E}\{C_t(x_{t-1}, \hat{D}_t)\},$$

where $\bar{C}_{t-1,t}(x_{t-1})$ is the expected costs that will be incurred during time interval t using the information known at time $t-1$.

5.3 MODELING RESOURCES

There is a vast range of problems that can be broadly described in terms of managing "resources." Resources can be equipment, people, money, robots or even games such as backgammon or chess. Depending on the setting, we might use the term asset (financial applications, expensive equipment) or entity (managing a robot, playing a board game). It is very common to start with fairly simple models of these problems, but the challenge is to solve complex problems.

There are four important problem classes that we consider in this volume, each offering unique computational challenges. These can be described along two dimensions: the number of resources or entities being managed, and the complexity of the attributes of each resource or entity. We may be managing a single entity (a robot, an aircraft, an electrical power plant) or multiple entities (a fleet of aircraft, trucks or locomotives, funds in different asset classes, groups of people). The attributes of each entity or resource may be simple (the truck may be described by its location, the money by the asset class into which it has been invested) or complex (all the characteristics of an aircraft or pilot).

The computational implications of each problem class can be quite different. Not surprisingly, different communities tend to focus on specific problem classes, making it possible for people to make the connection between mathematical notation (which can be elegant but vague) and the characteristics of a problem. Problems involving a single, simple entity can usually be solved using the classical techniques of Markov decision processes (chapter 3), although even here some problems can be difficult. The artificial intelligence community often works on problems involving a single, complex entity (games such as Connect-4 and backgammon, moving a robot arm or flying an aircraft). The operations research community has major subcommunities working on problems that involve multiple, simple entities (multicommodity flow problems, inventory problems), while portions of the simulation community and math programming community (for deterministic problems) will work on applications with multiple, complex entities.

In this section, we describe notation that allows us to evolve from simple to complex problems in a natural way. Our goal is to develop mathematical notation that does a better job of capturing which problem class we are working on.

5.3.1 Modeling a single, discrete resource

Many problems in dynamic programming involve managing a single resource such as flying an aircraft, planning a path through a network, or planning a game of chess or backgammon. These problems are distinctly different than those involving the management of fleets of vehicles, inventories of blood or groups of people. For this reason, we adopt specific notation for the single entity problem.

If we are managing a single, discrete resource, we find it useful to introduce specific notation to model the attributes of the resource. For this purpose, we use

$$\begin{aligned}
r_t &= \text{The attribute vector of the resource at time } t \\
&= (r_{t1}, r_{t2}, \dots, r_{tN}), \\
\mathcal{R} &= \text{Set of all possible attribute vectors.}
\end{aligned}$$

Attributes might be discrete (0, 1, 2, . . .), continuous ($0 \leq r_i \leq 1$) or categorical ($r_i =$ red). We typically assume that the number of dimensions of r_t is not too large. For example, if we are modeling the flow of product through a supply chain, the attribute vector might consist of the product type and location. If we are playing chess, the attribute vector would have 64 dimensions (the piece on each square of the board).

5.3.2 Modeling multiple resources

Imagine that we are modeling a fleet of unmanned aerial vehicles (UAV's), which are robotic aircraft used primarily for collecting information. We can let r_t be the attributes of a single UAV at time t, but we would like to describe the collective attributes of a fleet of UAV's. There is more than one way to do this, but one way that will prove to be notationally convenient is to define

$$\begin{aligned} R_{tr} &= \text{The number of resources with attribute } r \text{ at time } t, \\ R_t &= (R_{tr})_{r\in\mathcal{R}}. \end{aligned}$$

R_t is known as the *resource state vector*. If r is a vector, then $|\mathcal{R}|$ may be quite large. It is not hard to create problems where R_t has hundreds of thousands of dimensions. If elements of r_t are continuous, then in theory at least, R_t is infinite-dimensional. It is important to emphasize that in such cases, we would never enumerate the entire vector of elements in R_t.

We note that we can use this notation to model a single resource. Instead of letting r_t be our state vector (for the single resource), we let R_t be the state vector, where $\sum_{r\in\mathcal{R}} R_{tr} = 1$. This may seem clumsy, but it offers notational advantages we will exploit from time to time.

5.3.3 Illustration: the nomadic trucker

The "nomadic trucker" is a colorful illustration of a multiattribute resource which helps to illustrate some of the modeling conventions being introduced in this chapter. We use this example to illustrate different issues that arise in approximate dynamic programming, leading up to the solution of large-scale resource management problems later in our presentation.

The problem of the nomadic trucker arises in what is known as the truckload trucking industry. In this industry, a truck driver works much like a taxicab. A shipper will call a truckload motor carrier and ask it to send over a truck. The driver arrives, loads up the shipper's freight and takes it to the destination where it is unloaded. The shipper pays for the entire truck, so the carrier is not allowed to consolidate the shipment with freight from other shippers. In this sense, the trucker works much like a taxicab for people. However, as we will soon see, our context of the trucking company adds an additional level of richness that offers some relevant lessons for dynamic programming.

Our trucker runs around the United States, where we assume that his location is one of the 48 contiguous states. When he arrives in a state, he sees the customer demands for loads to move from that state to other states. There may be none, one, or several. He may choose a load to move if one is available; alternatively, he has the option of doing nothing or moving empty to another state (even if a load is available). Once he moves out of a state, all other customer demands (in the form of loads to be moved) are assumed to be picked up by other truckers and are therefore lost. He is not able to see the availability of loads out of states other than where he is located.

Although truckload motor carriers can boast fleets of over 10,000 drivers, our model focuses on the decisions made by a single driver. There are, in fact, thousands of trucking "companies" that consist of a single driver. In chapter 14 we will show that the concepts we develop here form the foundation for managing the largest and most complex versions of this problem. For now, our "nomadic trucker" represents a particularly effective way of illustrating some important concepts in dynamic programming.

A basic model

The simplest model of our nomadic trucker assumes that his only attribute is his location, which we assume has to be one of the 48 contiguous states. We let

$$\mathcal{I} = \text{The set of "states" (locations) at which the driver can be located.}$$

We use i and j to index elements of $\mathcal{I}$. His attribute vector then consists of

$$r = \{i\}.$$

In addition to the attributes of the driver, we also have to capture the attributes of the loads that are available to be moved. For our basic model, loads are characterized only by where they are going. Let

$$\begin{aligned} b &= \text{The vector of characteristics of a load} \\ &= \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} \text{The origin of the load.} \\ \text{The destination of the load.} \end{pmatrix}. \end{aligned}$$

We let $\mathcal{R}$ be the set of all possible values of the driver attribute vector r, and we let $\mathcal{B}$ be the set of all possible load attribute vectors b.

A more realistic model

We need a richer set of attributes to capture some of the realism of an actual truck driver. To begin, we need to capture the fact that at a point in time, a driver may be in the process of moving from one location to another. If this is the case, we represent the attribute vector as the attribute that we expect when the driver arrives at the destination (which is the next point at which we can make a decision). In this case, we have to include as an attribute the time at which we expect the driver to arrive.

Second, we introduce the dimension that the equipment may fail, requiring some level of repair. A failure can introduce additional delays before the driver is available.

A third important dimension covers the rules that limit how much a driver can be on the road. In the United States, drivers are governed by a set of rules set by the Department of Transportation ("DOT"). There are three basic limits: the amount a driver can be behind the wheel in one shift, the amount of time a driver can be "on duty" in one shift (includes time waiting), and the amount of time that a driver can be on duty over any contiguous eight-day period. As of this writing, these rules are as follows: a driver can drive at most 11 hours at a stretch, he may be on duty for at most 14 continuous hours (there are exceptions to this rule), and the driver can work at most 70 hours in any eight-day period. The last clock is reset if the driver is off-duty for 34 successive hours during any stretch (known as the "34-hour reset").

A final dimension involves getting a driver home. In truckload trucking, drivers may be away from home for several weeks at a time. Companies work to assign drivers that get them home in a reasonable amount of time.

If we include these additional dimensions, our attribute vector grows to

$$r_t = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \\ r_5 \\ r_6 \\ \\ r_7 \\ r_8 \end{pmatrix} = \begin{pmatrix} \text{The current or future location of the driver.} \\ \text{The time at which the driver is expected to arrive at his future location.} \\ \text{The maintenance status of the equipment.} \\ \text{The number of hours a driver has been behind the wheel during his} \\ \text{current shift.} \\ \text{The number of hours a driver has been on-duty during his current shift.} \\ \text{An eight-element vector giving the number of hours the driver was on} \\ \text{duty over each of the previous eight days.} \\ \text{The driver's home domicile.} \\ \text{The number of days a driver has been away from home.} \end{pmatrix}.$$

We note that element r_6 is actually a vector that holds the number of hours the driver was on duty during each calendar day over the last eight days.

A single attribute such as location (including the driver's domicile) might have 100 outcomes, or over 1,000. The number of hours a driver has been on the road might be measured to the nearest hour, while the number of days away from home can be as large as 30 (in rare cases). Needless to say, the number of potential attribute vectors is extremely large.

5.4 THE STATES OF OUR SYSTEM

The most important quantity in a dynamic program is the state variable. The state variable captures what we need to know, but just as important it is the variable around which we construct value function approximations. Success in developing a good approximation strategy depends on a deep understanding of what is important in a state variable to capture the future behavior of a system.

5.4.1 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, but offer a definition that appears to be somewhat tighter than what typically appears in the literature. We suggest the following definition:

Definition 5.4.1 *A* **state variable** *is the minimally dimensioned function of history that is necessary and sufficient to compute the decision function, the transition function, and the contribution function.*

Later in the chapter, we discuss the decision function (section 5.5), the transition function (section 5.7), and the objective function (section 5.8). In plain English, a state variable is all the information you need to know (at time t) to model the system from time t onward. Initially, it is easiest to think of the state variable in terms of the physical state of the system (the status of the pilot, the amount of money in our bank account), but ultimately it is necessary to think of it as the "state of knowledge."

This definition provides a very quick test of the validity of a state variable. If there is a piece of data in either the decision function, 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 "minimally dimensioned function" 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 this is not practical. 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 $(\hat{p}_1, \hat{p}_2, \dots, \hat{p}_t)$. 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 $(\hat{p}_t, \hat{p}_{t-1}, \dots, \hat{p}_{t-N+1})$ 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

$$S_t = (\hat{p}_t, \hat{p}_{t-1}, \dots, \hat{p}_{t-N+1}). \tag{5.4}$$

Many authors say that if we use the moving average model, we no longer have a proper state variable. Rather, we would have an example of a "history-dependent process" where the state variable needs to be augmented with history. Using our definition of a state variable, the concept of a history-dependent process has no meaning. The state variable is simply the minimal information required to capture what is needed to model future dynamics. Needless to say, having to explicitly retain history, as we did with the moving average model, produces a much larger state variable than the exponential smoothing model.

5.4.2 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 states:

The physical state - 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.

The information state - This encompasses the physical state as well as any other information we need to make a decision, compute the transition or compute the objective function.

The belief/knowledge state - If the information state is what we know, the belief state (also known as the knowledge state) captures how well we know it. This concept is largely absent from most dynamic programs, but arises in the setting of partially observable processes (when we cannot observe a portion of the state variable).

There are many problems in dynamic programming that involve the management of a single resource or entity (or asset - the best terminology depends on the context), such as using a computer to play backgammon, routing a single aircraft, controlling a power plant, or selling an asset. There is nothing wrong with letting S_t be the state of this entity. When we are managing multiple entities (which often puts us in the domain of "resource management"), it is often useful to distinguish between the state of a single entity, which we represent as r_t, versus the state of all the entities, which we represent as R_t.

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 or 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.

Alternatively, other information might include estimates of parameters of the system (costs, speeds, times, prices). To represent this, let

$$\begin{aligned}\bar{\theta}_t &= \text{A vector of estimates of different problem parameters at time } t.\\ \hat{\theta}_t &= \text{New information about problem parameters that arrive during time interval } t.\end{aligned}$$

We can think of $\bar{\theta}_t$ as the state of our information about different problem parameters at time t. We can now write a more general form of our state variable as:

$$\begin{aligned}S_t &= \text{The \textit{information state} at time } t\\ &= (R_t, \bar{\theta}_t).\end{aligned}$$

In chapter 12, we will show that it is important to include not just the point estimate $\bar{\theta}_t$, but the entire distribution (or the parameters needed to characterize the distribution, such as the variance).

A particularly important version of this more general state variable arises in approximate dynamic programming. Recall that in chapter 4 we used an approximation of the value function to make a decision, as in

$$x_t^n = \arg\max_{x_t \in \mathcal{X}_t^n} \left(C_t(R_t^n, x_t) + \overline{V}_t^{n-1}(R_t^x) \right) \tag{5.5}$$

Here $\overline{V}^{n-1}(\cdot)$ is an estimate of the value function if our decision takes us from resource state R_t to R_t^x, and x_t^n is the value of x_t that solves the right-hand side of (5.5). In this case, our state variable would consist of

$$S_t = (R_t, \overline{V}^{n-1}).$$

The idea that the value function is part of our state variable is quite important in approximate dynamic programming.

5.4.3 The post-decision state variable

We can view our system as evolving through sequences of new information followed by a decision followed by new information (and so on). Although we have not yet discussed decisions, for the moment let the decisions (which may be a vector) be represented generically using a_t (we discuss our choice of notation for a decision in the next section). In this case, a history of the process might be represented using

$$h_t = (S_0, a_0, W_1, a_1, W_2, a_2, \ldots, a_{t-1}, W_t).$$

h_t contains all the information we need to make a decision a_t at time t. As we discussed before, h_t is sufficient but not necessary. We expect our state variable to capture what is needed to make a decision, allowing us to represent the history as

$$h_t = (S_0, a_0, W_1, S_1, a_1, W_2, S_2, a_2, \ldots, a_{t-1}, W_t, S_t). \tag{5.6}$$

The sequence in equation (5.6) defines our state variable as occurring after new information arrives and before a decision is made. For this reason, we call S_t the *pre-decision state variable*. This is the most natural place to write a state variable because the point of capturing information from the past is to make a decision.

For most problem classes, we can design more effective computational strategies using the *post-decision state variable*. This is the state of the system after a decision a_t. For this reason, we denote this state variable S_t^a, which produces the history

$$h_t = (S_0, a_0, S_0^a, W_1, S_1, a_1, S_1^a, W_2, S_2, a_2, S_2^a, \ldots, a_{t-1}, S_{t-1}^a, W_t, S_t). \tag{5.7}$$

We again emphasize that our notation S_t^a means that this function has access to all the exogenous information up through time t, along with the decision a_t (which also has access to the information up through time t).

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

■ **EXAMPLE 5.1**

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 $S_t^a = (k)$ (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 5.2**

The nomadic trucker revisited. Let $R_{tr} = 1$ if the trucker has attribute vector r 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 $S_t^a = R_t^a$ where $R_{tr}^a = 1$ if the trucker has attribute vector r after a decision has been made.

■ **EXAMPLE 5.3**

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 a_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 importance of the post-decision state variable, and how to use it, depends on the problem at hand. We saw in chapter 4 that the post-decision state variable allowed us to make decisions without having to compute the expectation within the $\max$ or $\min$ operator. Later we will see that this allows us to solve some very large scale problems.

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

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

$$\begin{aligned} S_t^a &= S^{M,a}(S_t, a_t), && (5.8) \\ S_{t+1} &= S^{M,W}(S_t^a, W_{t+1}). && (5.9) \end{aligned}$$

The structure of these functions is highly problem-dependent. However, there are sometimes significant computational benefits, primarily when we face the problem of approximating the value function. Recall that the state variable captures all the information we need to make a decision, compute the transition function, and compute the contribution function. S_t^a *only* has to carry the information needed to compute the transition function. For some applications, S_t^a has the same dimensionality as S_t, but in many settings, S_t^a is dramatically simpler than S_t, simplifying the problem of approximating the value function.

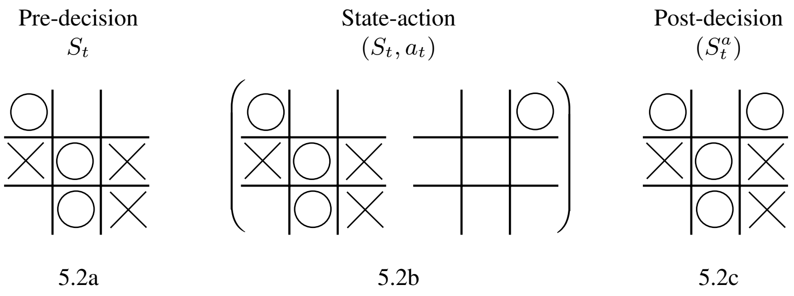


Figure 5.2 Pre-decision state, augmented state-action, and post-decision state for tic-tac-toe.

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

$$S_t^a = (S_t, a_t).$$

Figure 5.2 provides a nice illustration using our tic-tac-toe example. Figure 5.2a shows a tic-tac-toe board just before player O makes his move. Figure 5.2b shows the augmented state-action pair, where the decision (O decides to place his move in the upper right hand corner) is distinct from the state. Finally, figure 5.2c shows the post-decision state. For this example, the pre- and post-decision state spaces are the same, while the augmented state-action pair is nine times larger.

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

The augmented state variable is used in a popular class of algorithms known as Q-learning (introduced in chapter 6), where the challenge is to statistically estimate Q-factors which give the value of being in state S_t *and* taking action a_t. The Q-factors are written $Q(S_t, a_t)$, 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 action by solving $\min_a Q(S_t, a_t)$. This is the essence of Q-learning, but the price of this algorithmic step is that we have to estimate $Q(S_t, a_t)$ for each S_t and a_t. It is not possible to determine a_t by optimizing a function of S_t^a alone, since we generally cannot determine which action a_t brought us to S_t^a.

The post-decision as a point estimate Assume that we have a problem where we can compute a point estimate of future information. Let $\bar{W}_{t,t+1}$ 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 $\bar{W}_{t,t+1} = \mathbb{E}(W_{t+1}|S_t)$ or $\bar{W}_{t,t+1} = 0$. W_{t+1} might be a discrete outcome such as the number of equipment failures. It may not make sense to use an expectation (we may have problems working with 0.10 failures), so in this setting W_{t+1} might be the most likely outcome. Finally, we might simply assume that W_{t+1} is empty (a form of "null" field). For example, a taxi picking up a customer may not know the destination of the customer before the customer gets in the cab. In this case, if W_{t+1} represents the destination, we might use $\bar{W}_{t,t+1} =$ '-'.

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

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

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

5.4.4 Partially observable states*

There are many applications where we are not able to observe (or measure) the state of the system precisely, as illustrated in the examples. These problems are referred to as *partially observable Markov decision processes*, and require introducing a new class of exogenous information representing the difference between the true state and the observed state.

■ **EXAMPLE 5.1**

A retailer may have to order inventory without being able to measure the precise current inventory. It is possible to measure sales, but theft and breakage introduce errors.

■ **EXAMPLE 5.2**

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 5.3**

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.

To model this class of applications, let

$\tilde{S}_t$	$=$	The true state of the system at time t.
$\widetilde{W}_t$	$=$	Errors that arise when measuring the state $\tilde{S}_t$.

In this context, we assume that our state variable S_t is the observed state of the system. Now, our history is given by

$$\begin{aligned} h_t = \; & (S_0, a_0, S_0^a, W_1, \tilde{S}_1, \widetilde{W}_1, S_1, a_1, S_1^a, W_2, \tilde{S}_2, \widetilde{W}_2, S_2, a_2, S_2^a, \ldots, \\ & a_{t-1}, S_{t-1}^a, W_t, \tilde{S}_t, \widetilde{W}_t, S_t). \end{aligned}$$

We view our original exogenous information W_t as representing information such as the change in price of a resource, a customer demand, equipment failures, or delays in the completion of a task. By contrast, $\widetilde{W}_t$, which captures the difference between $\tilde{S}_t$ and S_t, represents measurement error or the inability to observe information. Examples of measurement error might include differences between actual and calculated inventory of product on a store shelf (due, for example, to theft or breakage), the error in estimating the location of a vehicle (due to errors in the GPS tracking system), or the difference between the actual state of a piece of machine such as an aircraft (which might have a failed part) and the observed state (we do not yet know about the failure). A different form of observational error arises when there are elements we simply cannot observe (for example, we know the location of the vehicle but not its fuel status).

It is important to realize that there are two transition functions at work here. The "real" transition function models the dynamics of the true (unobservable) state, as in

$$\tilde{S}_{t+1} = \tilde{S}^M(\tilde{S}_t, a_t, \widetilde{W}_{t+1}).$$

In practice, not only do we have the problem that we cannot perfectly measure $\tilde{S}_t$, we may not know the transition function $\tilde{S}^M(\cdot)$. Instead, we are working with our "engineered" transition function

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

where W_{t+1} is capturing some of the effects of the observation error. When building a model where observability is an issue, it is important to try to model $\tilde{S}_t$, the transition function $\tilde{S}^M(\cdot)$ and the observation error $\widetilde{W}_t$ as much as possible. However, anything we cannot measure may have to be captured in our generic noise vector W_t.

5.4.5 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. This is the style we used in chapters 3 and 4.

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 very compact models that have been popular in the operations research community. The presentation in chapter 3 depends on this representation. However, the use of a single index completely disguises the structure of the state variable, and often produces intractably large state spaces.

In the arena of approximate dynamic programming, 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.

Our attribute vector notation, which we use to describe a single entity, is an example of a factored representation. Each element r_i of an attribute vector represents a particular feature of the entity. The resource state variable $R_t = (R_{tr})_{r\in\mathcal{R}}$ is also a factored representation, since we explicitly capture the number of resources with a particular attribute. This is useful when we begin developing approximations for problems such as the dynamic assignment problem that we introduced in section 2.2.10.

5.5 MODELING DECISIONS

Fundamental to dynamic programs is the characteristic that we are making decisions over time. For stochastic problems, we have to model the sequencing of decisions and information, but there are many applications of dynamic programming that address deterministic problems.

Virtually all problems in approximate dynamic programming have large state spaces; it is hard to devise a problem with a small state space which is hard to solve. But problems can vary widely in terms of the nature of the decisions we have to make. In this section, we introduce two notational styles (which are illustrated in chapters 2 and 4) to help communicate the breadth of problem classes.

5.5.1 Decisions, actions, and controls

A survey of the literature reveals a distressing variety of words used to mean "decisions." The classical literature on Markov decision process talks 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) or a policy (a rule for choosing an action). 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, and the simulation community wants to apply a rule.

The proper notation for decisions will depend on the specific application. Rather than use one standard notation, we use two. Following the widely used convention in the reinforcement learning community, we use a whenever we have a finite number of discrete actions. The action space $\mathcal{A}$ might have 10, 100 or perhaps even 1,000 actions, but we are not aware of actual applications with, say, 10,000 actions or more.

There are many applications where a decision is either continuous or vector-valued. For example, in chapters 2 and 14 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* (enumerating the number of potential actions, even if x_t is discrete, is meaningless). These high-dimensional decision vectors arise frequently in the types of resource allocation problems addressed in operations research.

There is an obvious equivalence between the "a" notation and the "x" notation. Let $d = a$ represent the decision to make a decision of "type" a. Then $x_d = 1$ corresponds to the decision to take action a. We would like to argue that one representation is better than the other, but the simple reality is that both are useful. We could simply stay with the "a" notation, allowing a to be continuous or a vector, as needed. However, there are many algorithms, primarily developed in the reinforcement learning community, where we really need to insist that the action space be finite, and we feel it is useful to let the use of a for action communicate this property. By contrast, when we switch to x notation, we are communicating that we can no longer enumerate the action space, and have to turn to other types of search algorithms to find the best action. In general, this means that we cannot use lookup table approximations for the value function, as we did in chapter 4.

It is important to realize that problems with small, discrete action spaces, and problems with continuous and/or multidimensional decisions, both represent important problem classes, and both can be solved using the algorithmic framework of approximate dynamic programming.

5.5.2 Making decisions

The challenge of dynamic programming is making decisions. In a deterministic setting, we can pose the problem as one of making a sequence of decisions $a_0, a_1, \dots, a_T$ (if we are lucky enough to have a finite-horizon problem). However, in a stochastic problem the challenge is finding the best *policy* for making a decision.

It is common in the dynamic programming community to represent a policy by π. Given a state S_t, an action would be given by $a_t = \pi(S_t)$ or $x_t = \pi(S_t)$ (depending on our notation). We feel that this notation can be a bit abstract, and also limits our ability to specify classes of policies. Instead, we prefer to emphasize that a policy is, in fact, a *function* which returns a decision. As a result, we let $a_t = A^\pi(S_t)$ (or $x_t = X^\pi(S_t)$) represent the function (or decision function) that returns an action a_t given state S_t. We use π as a superscript to capture the fact that $A^\pi(S_t)$ is one element in a family of functions which we represent by writing $\pi \in \Pi$.

We refer to $A^\pi(S_t)$ (or $X^\pi(S_t)$) and π interchangeably as a policy. A policy can be a simple function. For example, in an inventory problem, let S_t be the number of items that we have in inventory. We might use a reorder policy of the form

$$A(S_t) = \begin{cases} Q - S_t & \text{if } S_t < q, \\ 0 & \text{otherwise.} \end{cases}$$

We might let Π^{OUT} be the set of "order-up-to" decision rules. Searching for the best policy $\pi \in \Pi^{OUT}$ means searching for the best set of parameters (Q, q). Finding the best policy within a particular set does not mean that we are finding the optimal policy, but in many cases that will be the best we can do.

Finding good policies is the challenge of dynamic programming. We provided a glimpse of how to do this in chapter 4, but we defer to chapter 6 a more thorough discussion of policies, which sets the tone for the rest of the book.

5.6 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. While there are many important deterministic dynamic programs, exogenous information processes represent an important dimension in many problems in resource management.

5.6.1 Basic notation for information processes

The central challenge of dynamic programs is dealing with one or more exogenous information processes, forcing us to make decisions before all the information is known. These might be stock prices, travel times, equipment failures, or the behavior of an opponent in a

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

game. There might be a single exogenous process (the price at which we can sell an asset) or a number of processes. For a particular problem, we might model this process using notation that is specific to the application. Here, we introduce generic notation that can handle any problem.

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

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

Here, $\hat{p}_{t+1}$ is an exogenous random variable representing the chance 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. For example, we might assume that it is normally distributed 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 5.1 shows 10 sample realizations of a price process that starts with $p_0 = 29.80$ but then evolves according to the sample realization. Following standard mathematical 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

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

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

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

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 $\hat{\Omega}$.

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

$$W_t = \text{The exogenous information becoming available during interval } t.$$

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.

The choice of notation W_t as a generic "information function" is not standard, but it is mnemonic (it looks like ω_t). We would then write $\omega_t = W_t(\omega)$ as a sample realization of the information arriving during time interval t. This notation adds a certain elegance when we need to write decision functions and information in the same equation.

Some authors use ω to index a particular sample path, where $W_t(\omega)$ is the information that arrives during time interval t. Other authors view ω as the information itself, as in

$$\omega = (-0.24, 2.25, 1.48).$$

Obviously, both are equivalent. Sometimes it is convenient to define

$$\begin{aligned}
\omega_t &= \text{The information that arrives during time period } t \\
&= W_t(\omega), \\
\omega &= (\omega_1, \omega_2, \ldots).
\end{aligned}$$

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

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

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.

5.6.2 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). The term scenario causes problems of both notation and interpretation. First, "scenario" and "state" create an immediate competition for the interpretation of the letter "s." Second, "scenario" is often used in the context of major events. For example, we can talk about the scenario that the Chinese might revalue their currency (a major question in the financial markets at this time). We could talk about two scenarios: (1) the Chinese hold the current relationship between the yuan and the dollar, and (2) they allow their currency to float. For each scenario, we could talk about the fluctuations in the exchange rates between all currencies.

Recognizing that different communities use "outcome" and "scenario" to mean the same thing, we suggest that we may want to reserve the ability to use both terms simultaneously. For example, we might have a set of scenarios that determine if and when the Chinese revalue their currency (but this would be a small set). 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 (currency exchange rates, for example).

■ **EXAMPLE 5.1**

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 (failure rate curve), ω represents a sample realization of failures (transformers can fail at any time, although the likelihood they will fail depends on ψ).

■ **EXAMPLE 5.2**

Energy resource planning - The federal government has to determine energy sources that will replace fossil fuels. As research takes place, there are random improvements in various technologies. However, the planning of future energy technologies depends on the success of specific options, notably whether we will be able to sequester carbon underground. If this succeeds, we will be able to take advantage of vast stores of coal in the United States. Otherwise, we have to focus on options such as hydrogen and nuclear.

In section 13.7, we provide a brief introduction to the field of stochastic programming where "scenario" is the preferred term to describe a set of random outcomes. Often, applications of stochastic programming apply to problems where the number of outcomes (scenarios) is relatively small.

5.6.3 Lagged information processes*

There are many settings where the information about a new arrival comes before the new arrival itself as illustrated in the examples.

■ **EXAMPLE 5.1**

An airline may order an aircraft at time t and expect the order to be filled at time t'.

■ **EXAMPLE 5.2**

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

■ **EXAMPLE 5.3**

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

This concept is important enough that we offer the following term:

Definition 5.6.1 *The actionable time of a resource is the time at which a decision may be used to change its attributes (typically generating a cost or reward).*

The actionable time is simply one attribute of a resource. For example, if at time t we own a set of futures purchased in the past with exercise dates of $t+1, t+2, \dots, t'$, then the exercise date would be an attribute of each futures contract (the exercise dates do not need to coincide with the discrete time instances when decisions are made). When writing out a mathematical model, it is sometimes useful to introduce an index just for the actionable time (rather than having it buried as an element of the attribute vector a). Before, we let R_{ta} be the number of resources that we know about at time t with attribute vector a. The attribute might capture that the resource is not actionable until time t' in the future. If we need to represent this explicitly, we might write

$$\begin{aligned} R_{t,t'a} &= \text{The number of resources that we know about at time } t \text{ that will} \\ &\quad\ \text{be actionable with attribute vector } a \text{ at time } t', \\ R_{tt'} &= (R_{t,t'a})_{a\in\mathcal{A}}, \\ R_t &= (R_{t,t'})_{t'\geq t}. \end{aligned}$$

It is very important to emphasize that while t is discrete (representing when decisions are made), the actionable time t' may be continuous. When this is the case, it is generally best to simply leave it as an element of the attribute vector.

5.6.4 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

A large number of problems involve processes that evolve independently of the state of the system, such as wind (in an energy application), stock prices (in the context of small trading decisions) and demand for a product (assuming inventories are small relative to the market).

■ **EXAMPLE 5.1**

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 5.2**

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-dependent information processes

The standard dynamic programming models allow the probability distribution of new information (for example, the chance in price of an asset) to be a function of the state of the system (the mean change in the price might be negative if the price is high enough, positive if the price is low enough). This is a more general model than one with independent increments, where the distribution is independent of the state of the system.

■ **EXAMPLE 5.1**

Customers arrive at an automated teller machine according to a Poisson process, but as the line grows longer, an increasing proportion decline to join the queue (a property known as *balking* in the queueing literature). The apparent arrival rate at the queue is a process that depends on the length of the queue.

■ **EXAMPLE 5.2**

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. Conversely, upward movement may be interpreted as a signal that people are buying the stock, encouraging more buying behavior.

Interestingly, many models of Markov decision processes use information processes that do, in fact, exhibit independent increments. For example, we may have a queueing problem where the state of the system is the number of customers in the queue. The number of arrivals may be Poisson, and the number of customers served in an increment of time is determined primarily by the length of the queue. It is possible, however, that our arrival process is a function of the length of the queue itself (see the examples for illustrations).

State-dependent information processes are more difficult to model and introduce additional parameters that must be estimated. However, from the perspective of dynamic programming, they do not introduce any fundamental complexities. As long as the distribution of outcomes is dependent purely on the state of the system, we can apply our standard models. In fact, approximate dynamic programming algorithms simply need some mechanism to sample information. It does not even matter if the exogenous information depends on information that is not in the state variable (although this will introduce errors).

It is also possible that the information arriving to the system depends on its state, as depicted in the next set of examples.

■ **EXAMPLE 5.1**

A driver is planning a path over a transportation network. When the driver arrives at intersection i of the network, he is able to determine the transit times of each of the segments (i, j) emanating from i. Thus, the transit times that are observed by the driver depend on the path taken by the driver.

■ **EXAMPLE 5.2**

A private equity manager learns information about a company only by investing in the company and becoming involved in its management. The information arriving to the manager depends on the state of his portfolio.

This is a different form of state-dependent information process. Normally, an outcome ω is assumed to represent *all* information available to the system. A probabilist would insist that this is still the case with our driver; the fact that the *driver* does not know the transit times on all the links is simply a matter of modeling the information the driver uses. However, many will find it more natural to think of the information as depending on the state.

Action-dependent information processes

Imagine that a mutual fund is trying to optimize the process of selling a large position in a company. If the mutual fund makes the decision to sell a large number of shares, the effect may be to depress the stock price because the act of selling sends a negative signal to the market. Thus, the action may influence what would normally be an exogenous stochastic process.

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* can be thought of as a probability density function $\phi_t(\omega_t)$ that gives the density (we would say the probability of ω if it were discrete) of an outcome ω_t in time t. If the problem has independent increments, we would write the density simply as $\phi_t(\omega_t)$. If the information process is Markovian (dependent on a state variable), then we would write it as $\phi_t(\omega_t|S_{t-1})$.

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.

5.6.5 Supervisory processes*

We are often 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 5.1**

Consider our nomadic trucker where we measure his state s^n (his state) and his decision a^n which we represent in terms of the destination of his next move. We could use a historical file $(s^m, a^m)_{m=1}^n$ to build a probability distribution $\rho(s, a)$ which gives the probability that we make decision a given his state s. We can use $\rho(s, a)$ to predict decisions in the future.

■ **EXAMPLE 5.2**

A mutual fund manager adds x_t dollars in cash at the end of day t (to be used to cover withdrawals on day $t + 1$) when there are R_t dollars in cash left over at the end of the day. We can use a series of observations of x_{t_m} and R_{t_m} on days $t_1, t_2, \ldots, t_m$ to fit a model of the form $X(R) = \theta_0 + \theta_1 R + \theta_2 R^2 + \theta_3 R^3$.

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 an approximate dynamic programming algorithm to help fit value functions that can be used to improve the decision function. The supervisory process helps provide an initial policy that may not be perfect, but at least is reasonable.

5.6.6 Policies in the information process*

The sequence of information $(\omega_1, \omega_2, \ldots, \omega_t)$ is assumed to be driven by some sort of exogenous process. However, we are generally interested in quantities that are functions of both exogenous information as well as the decisions. It is useful to think of decisions as *endogenous information*. But where do the decisions come from? We now see that decisions come from policies. In fact, it is useful to represent our sequence of information and decisions as

$$H_t^\pi = (S_0, X_0^\pi, W_1, S_1, X_1^\pi, W_2, S_2, X_2^\pi, \ldots, X_{t-1}^\pi, W_t, S_t). \tag{5.10}$$

Now our history is characterized by a family of functions: the information variables W_t, the decision functions (policies) X_t^π, and the state variables S_t. We see that to characterize a particular history h_t, we have to specify both the sample outcome ω as well as the policy π. Thus, we might write a sample realization as

$$h_t^\pi = H_t^\pi(\omega).$$

We can think of a complete history $H_\infty^\pi(\omega)$ as an outcome in an expanded probability space (if we have a finite horizon, we would denote this by $H_T^\pi(\omega)$). Let

$$\omega^\pi = H_\infty^\pi(\omega)$$

be an outcome in our expanded space, where ω^π is determined by ω and the policy π. Let Ω^π be the set of all outcomes of this expanded space. The probability of an outcome in Ω^π obviously depends on the policy we are following. Thus, computing expectations (for example, expected costs or rewards) requires knowing the policy as well as the set of exogenous outcomes. For this reason, if we are interested, say, in the expected costs during time period t, some authors will write $E_t^\pi\{C_t(S_t, x_t)\}$ to express the dependence of the expectation on the policy. However, even if we do not explicitly index the policy, it is important to understand that we need to know how we are making decisions if we are going to compute expectations or other quantities.

5.7 THE TRANSITION FUNCTION

The next step in modeling a dynamic system is the specification of the *transition function*. This function describes how the system evolves from one state to another as a result of decisions and information. 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.

5.7.1 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 dynamics of a system can be represented in different ways. The easiest is through a simple function that works as follows

$$S_{t+1} = S^M(S_t, X_t^\pi, W_{t+1}). \tag{5.11}$$

The function $S^M(\cdot)$ 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," "transition law," and "transition function." We prefer "transition function" because it is the most descriptive. We use the notation $S^M(\cdot)$ to reflect that this is the *state* transition function, which represents a *model* of the dynamics of the system. Below, we reinforce the "M" superscript with other modeling devices.

The arguments of the function follow standard notational conventions in the control literature (state, action, information), but different authors will follow one of two conventions for modeling time. While equation (5.11) is fairly common, many authors will write the recursion as

$$S_{t+1} = S^M(S_t, X_t^\pi, W_t). \tag{5.12}$$

If we use the form in equation (5.12), we would say "the state of the system at the beginning of time interval $t+1$ is determined by the state at time t, plus the decision that is made at time t and the information that arrives during time interval t." In this representation, t indexes when we are using the information. We refer to (5.12) as the *actionable representation* since it captures when we can act on the information. This representation is always used for deterministic models, and many authors adopt it for stochastic models as well. We prefer the form in equation (5.11) since we are measuring the information available at time t when we are about to make a decision. If we are making a decision x_t at time t, it is natural to index by t all the variables that can be measured at time t. We refer to this style as the *informational representation*.

In equation (5.11), we have written the function assuming that the *function* does not depend on time (it does depend on data that depends on time). A common notational error is to write a function, say, $S_t^M(S_t, x_t)$ as if it depends on time, when in fact the function is stationary, but depends on data that depends on time. If the parameters (or structure) of the function depend on time, then we would use $S_t^M(S_t, x_t, W_{t+1})$ (or possibly $S_{t+1}^M(S_t, x_t, W_{t+1})$). If not, the transition function should be written $S^M(S_t, x_t, W_{t+1})$.

This is a very general way of representing the dynamics of a system. In many problems, the information W_{t+1} arriving during time interval $t+1$ depends on the state S_t at the end of time interval t, but is conditionally independent of all prior history given S_t. For example, a driver moving over a road network may only learn about the travel times on a link from i to j when he arrives at node i. When this is the case, we say that we have a Markov information process. When the decisions depend only on the state S_t, then we have a Markov decision process. In this case, we can store the system dynamics in the form of a one-step transition matrix using

$P(s'\|s,x)$	$=$	The probability that $S_{t+1} = s'$ given $S_t = s$ and $X_t^\pi = x$,
P^π	$=$	Matrix of elements where $P(s'\|s,x)$ is the element in row s and column s' and where the decision x to be made in each state is determined by a policy π.

There is a simple relationship between the transition function and the one-step transition matrix. Let

$$1_X = \begin{cases} 1 & X \text{ is true} \\ 0 & \text{Otherwise.} \end{cases}$$

Assuming that the set of outcomes Ω is discrete, the one-step transition matrix can be computed using

$$\begin{aligned} P(s'|s,x) &= \mathbb{E}\{1_{\{s'=S^M(S_t,x,W_{t+1})\}}|S_t = s\} \\ &= \sum_{\omega_{t+1}\in\Omega_{t+1}} P(W_{t+1} = \omega_{t+1}) 1_{\{s'=S^M(S_t,x,W_{t+1})\}}. \end{aligned} \tag{5.13}$$

It is common in the field of Markov decision processes to assume that the one-step transition is given as data. Often, it can be quickly derived (for simple problems) using assumptions about the underlying process. For example, consider a financial asset selling problem with state variable $S_t = (R_t, p_t)$ where

$$R_t = \begin{cases} 1 & \text{We are still holding the asset,} \\ 0 & \text{The asset has been sold.} \end{cases}$$

and where p_t is the price at time t. We assume the price process is described by

$$p_t = p_{t-1} + \epsilon_t,$$

where ϵ_t is a random variable with distribution

$$\epsilon_t = \begin{cases} +1 & \text{with probability 0.3,} \\ 0 & \text{with probability 0.6,} \\ -1 & \text{with probability 0.1.} \end{cases}$$

Assume the prices are integer and range from 1 to 100. We can number our states from 0 to 100 using

$$\mathcal{S} = \{(0,-),(1,1),(1,2),\ldots,(1,100)\}.$$

We propose that our rule for determining when to sell the asset is of the form

$$X^\pi(R_t,p_t) = \begin{cases} \text{Sell asset} & \text{if } p_t < \bar{p}, \\ \text{Hold asset} & \text{if } p_t \geq \bar{p}. \end{cases}$$

Assume that $\bar{p} = 60$. A portion of the one-step transition matrix for the rows and columns corresponding to the state $(0,-)$ and $(1,58),(1,59),(1,60),(1,61),(1,62)$ looks like

$$P^{60} = \begin{array}{c} \text{(0,-)} \\ \text{(1,58)} \\ \text{(1,59)} \\ \text{(1,60)} \\ \text{(1,61)} \\ \text{(1,62)} \end{array} \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .1 & .6 & .3 & 0 \\ 0 & 0 & 0 & .1 & .6 & .3 \\ 0 & 0 & 0 & 0 & .1 & .6 \end{array} \right].$$

As we saw in chapter 3, this matrix plays a major role in the theory of Markov decision processes, although its value is more limited in practical applications. By representing the system dynamics as a one-step transition matrix, it is possible to exploit the rich theory surrounding matrices in general and Markov chains in particular.

In engineering problems, it is far more natural to develop the transition function first. Given this, it may be possible to compute the one-step transition matrix exactly or estimate it using simulation. The techniques in this book do not, in general, use the one-step transition matrix, but instead use the transition function directly. However, formulations based on the transition matrix provide a powerful foundation for proving convergence of both exact and approximate algorithms.

5.7.2 The resource transition function

There are many dynamic programming problems can be modeled in terms of managing "resources" where the state vector is denoted R_t. We use this notation when we want to specifically exclude other dimensions that might be in a state variable (for example, the challenge of making decisions to better estimate a quantity, which was first introduced in section 2.3). If we are using R_t as the state variable, the general representation of the transition function would be written

$$R_{t+1} = R^M(R_t, x_t, W_{t+1}).$$

Our notation is exactly analogous to the notation for a general state variable S_t, but it opens the door to other modeling dimensions. For now, we illustrate the resource transition equation using some simple applications.

Resource acquisition I - Purchasing resources for immediate use
Let R_t be the quantity of a single resource class we have available at the end of a time period, but before we have acquired new resources (for the following time period). The resource may be money available to spend on an election campaign, or the amount of oil, coal, grain or other commodities available to satisfy a market. Let $\hat{D}_t$ be the demand for the resource that occurs over time interval t, and let x_t be the quantity of the resource that is acquired at time t to be used during time interval $t+1$. The transition function would be written

$$R_{t+1} = \max\{0, R_t + x_t - \hat{D}_{t+1}\}.$$

Resource acquisition II: purchasing futures
Now assume that we are purchasing futures at time t to be exercised at time $t' > t$. At the end of time period t, we would let $R_{tt'}$ be the number of futures we are holding that can be exercised during time period t' (where $t' > t$). Now assume that we purchase $x_{tt'}$ additional futures to be used during time period t'. Our system dynamics would look like

$$R_{t+1,t'} = \begin{cases} \max\{0, (R_{tt} + R_{t,t+1}) + x_{t,t+1} - \hat{D}_{t+1}\}, & t' = t+1, \\ R_{tt'} + x_{tt'}, & t' \geq t+2\ . \end{cases}.$$

In many problems, we can purchase resources on the spot market, which means we are allowed to see the actual demand before we make the decision. This decision would be represented by $x_{t+1,t+1}$, which means the amount purchased using the information that arrived during time interval $t+1$ to be used during time interval $t+1$ (of course, these decisions are usually the most expensive). In this case, the dynamics would be written

$$R_{t+1,t'} = \begin{cases} \max\{0, (R_{tt} + R_{t,t+1}) + x_{t,t+1} + x_{t+1,t+1} - \hat{D}_{t+1}\}, & t' = t+1, \\ R_{tt'} + x_{tt'}, & t' \geq t+2\,. \end{cases}$$

Planning a path through college

Consider a student trying to satisfy a set of course requirements (for example, number of science courses, language courses, departmentals, and so on). Let R_{tc} be the number of courses taken that satisfy requirement c at the end of semester t. Let x_{tc} be the number of courses the student enrolled in at the end of semester t for semester $t+1$ to satisfy requirement c. Finally let $\hat{F}_{tc}(x_{t-1})$ be the number of courses in which the student received a failing grade during semester t given x_{t-1}. This information depends on x_{t-1} since a student cannot fail a course that she was not enrolled in. The system dynamics would look like

$$R_{t+1,c} = R_{t,c} + x_{t,c} - \hat{F}_{t+1,c}.$$

Playing backgammon

In backgammon, there are 24 points on which pieces may sit, plus a bar when you bump your opponent's pieces off the board, plus the state that a piece has been removed from the board (you win when you have removed all of your pieces). Let $i \in \{1, 2, \ldots, 26\}$ represent these positions. Let R_{ti} be the number of your pieces on location i at time t, where $R_{ti} < 0$ means that your opponent has $|R_{ti}|$ pieces on i. Let $x_{tii'}$ be the number of pieces moved from point i to i' at time t. If $\omega_t = W_t(\omega)$ is the outcome of the roll of the dice, then the allowable decisions can be written in the general form $x_t \in \mathcal{X}_t(\omega)$ where the feasible region $\mathcal{X}_t(\omega)$ captures the rules on how many pieces can be moved given $W_t(\omega)$. For example, if we only have two pieces on a point, $\mathcal{X}_t$ would restrict us from moving more than two pieces from this point. Let δx_t be a column vector with element i given by

$$\delta x_{ti} = \sum_{i'} x_{ti'i} - \sum_{i''} x_{tii''}.$$

Now let $\hat{y}_{t+1}$ be a variable similar to x_t representing the moves of the opponent after we have finished our moves, and let $\delta\hat{y}_{t+1}$ be a column vector similar to δx_t. The transition equation would look like

$$R_{t+1} = R_t + \delta x_t + \delta\hat{y}_{t+1}.$$

A portfolio problem

Let R_{tk} be the amount invested in asset $k \in \mathcal{K}$ where $\mathcal{K}$ may be individual stocks, mutual funds or asset classes such as bonds and money market instruments. Assume that each month we examine our portfolio and shift money from one asset to another. Let $x_{tkk'}$ be the amount we wish to move from asset k to k', where x_{tkk} is the amount we hold in asset k. We assume the transition is made instantly (the issue of transaction costs are not relevant here). Now let $\hat{\rho}_{t+1,k}$ be the return for asset k between t and $t+1$. The transition equation would be given by

$$R_{t+1,k} = \hat{\rho}_{t+1,k}\left(\sum_{k' \in \mathcal{K}} x_{tk'k}\right).$$

$$\begin{array}{ccc} \text{t=40} & \text{t=40} & \text{t=60} \\ \text{pre-decision} & \text{post-decision} & \text{pre-decision} \\ \begin{pmatrix} St.Louis \\ 41.4 \end{pmatrix} & \begin{pmatrix} LosAngeles \\ 65.0 \end{pmatrix} & \begin{pmatrix} LosAngeles \\ 70.4 \end{pmatrix} \end{array}$$

Figure 5.3 Pre- and post-decision attributes for a nomadic trucker.

We note that it is virtually always the case that the returns $\hat{\rho}_{tk}$ are correlated across the assets. When we use a sample realization $\hat{\rho}_{t+1}(\omega)$, we assume that these correlations are reflected in the sample realization.

5.7.3 Transition functions for complex resources*

When we are managing multiple, complex resources, each of which are described by an attribute vector r, it is useful to adopt special notation to describe the evolution of the system. Recall that the state variable S_t might consist of both the resource state variable R_t as well as other information. We need notation that specifically describes the evolution of R_t separately from the other variables.

We define the *attribute transition function* which describes how a specific entity with attribute vector r is modified by a decision of type a. This is modeled using

$$r_{t+1} = r^M(r_t, a_t, W_{t+1}). \tag{5.14}$$

The function $r^M(\cdot)$ parallels the state transition function $S^M(\cdot)$, but it works at the level of a decision of type a acting on a resource of type r. It is possible that there is random information affecting the outcome of the decision. For example, in section 5.3.3, we introduced a realistic version of our nomadic trucker where the attributes include dimensions such as the estimated time that the driver will arrive in a city (random delays can change this), and the maintenance status of the equipment (the driver may identify an equipment problem while moving).

As with our state variable, we let r_t be the attribute just before we make a decision. Although we are acting on the resource with a decision of type a, we retain our notation for the post-decision state and let r_t^a be the post-decision attribute vector. A simple example illustrates the pre- and post-decision attribute vectors for our nomadic trucker. Assume our driver has two attributes: location and the expected time at which he can be assigned to a new activity. Figure 5.3 shows that at time $t = 40$, we expect the driver to be available in St. Louis at time $t = 41.4$. At $t = 40$, we make the decision that as soon as the driver is available, we are going to assign him to a load going to Los Angeles, where we expect him (again at $t = 40$) to arrive at time $t = 65.0$. At time $t = 60$, he is still expected to be heading to Los Angeles, but we have received information that he has been delayed and now expect him to be available at time $t = 70.4$ (the delay is the new information).

As before, we can break down the effect of decisions and information using

$$\begin{aligned} r_t^a &= r^{M,a}(r_t, a_t), \\ r_{t+1} &= r^{M,W}(r_t^a, W_{t+1}). \end{aligned}$$

For algebraic purposes, it is also useful to define the indicator function

$$\begin{aligned}
\delta^a_{r'}(r,a) &= \begin{cases} 1, & r^a_t = r' = r^{M,a}(r_t,a_t), \\ 0 & \text{otherwise.} \end{cases} \\
\Delta^a &= \text{Matrix with } \delta^a_{r'}(r,a) \text{ in row } r' \text{ and column } (r,a).
\end{aligned}$$

The function $\delta^a(\cdot)$ (or matrix Δ^a) gives the post-decision attribute vector resulting from a decision a (in the case of Δ^a, a set of decisions represented by a_t).

It is convenient to think of acting on a single resource with an action a. If we have multiple resources (or types of resources), it also helps to let x_{tra} be the number of types we act on a resource of type r with action a, and let x_t be the vector of these decisions. We can then describe the evolution of an entire set of resources, represented by the vector R_t, using

$$R_{t+1} = R^M(R_t, x_t, W_{t+1}). \tag{5.15}$$

In this setting, we would represent the random information W_{t+1} as exogenous changes to the resource vector, given by

$\hat{R}_{t+1,r}$ = The change in the number of resources with attribute vector r due to information arriving during time interval $t+1$.

We can now write out the transition equation using

$$R_{t+1,r'} = \sum_{r\in\mathcal{R}} \sum_{a\in\mathcal{A}} \delta^a_{r'}(r,a) x_{tra} + \hat{R}_{t+1,r'}$$

or in matrix-vector form

$$R_{t+1} = \Delta^x R_t + \hat{R}_{t+1}.$$

5.7.4 Some special cases*

It is important to realize when special cases offer particular modeling challenges (or opportunities). Below we list a few we have encountered.

Deterministic resource transition functions

It is quite common in resource allocation problems that the attribute transition function is deterministic (the equipment never breaks down, there are no delays, resources do not randomly enter and leave the system). The uncertainty may arise not in the evolution of the resources that we are managing, but in the "other information" such as customer demands (for example, the loads that the driver might move) or our estimates of parameters (where we use our decisions to collect information on random variables with unknown distributions). If the attribute transition function is deterministic, then we would write $r^a_t = r^M(r_t, a_t)$ and we would have that $r_{t+1} = r^a_t$. In this case, we simply write

$$r_{t+1} = r^M(r_t, a_t).$$

This is an important special case since it arises quite often in practice. It does not mean the transition function is deterministic. For example, a driver moving over a network faces the decision, at node i, whether to go to node j. If he makes this decision, he will go to node j deterministically, but the cost or time over the link from i to j may be random.

Another example arises when we are managing resources (people, equipment, blood) to serve demands that arrive randomly over time. The effect of a decision acting on a resource is deterministic. The only source of randomness is in the demands that arise in each time period. Let R_t be the vector describing our resources, and let $\hat{D}_t$ be the demands that arose during time interval t. The state variable is $S_t = (R_t, \hat{D}_t)$ where $\hat{D}_t$ is purely exogenous. If x_t is our decision vector which determines which resources are assigned to each demand, then the resource transition function $R_{t+1} = R^M(R_t, x_t)$ would be deterministic.

Gaining and losing resources

In addition to the attributes of the modified resource, we sometimes have to capture the fact that we may gain or lose resources in the process of completing a decision. We might define

$\rho_{t+1,r,a}$	$=$	The multiplier giving the quantity of resources with attribute vector r available after being acted on with decision a at time t.

The multiplier may depend on the information available at time t (in which case we would write it as ρ_{tra}), but is often random and depends on information that has not yet arrived (in which case we use $\rho_{t+1,r,a}$). Illustrations of gains and losses are given in the next set of examples.

■ **EXAMPLE 5.1**

A corporation is holding money in an index fund with a 180-day holding period (money moved out of this fund within the period incurs a four percent load) and would like to transfer them into a high yield junk bond fund. The attribute of the resource would be $r = (\text{Type}, \text{ Age})$. There is a transaction cost (the cost of executing the trade) and a gain ρ, which is 1.0 for funds held more than 180 days, and 0.96 for funds held less than 180 days.

■ **EXAMPLE 5.2**

Transportation of liquefied natural gas - A company would like to purchase 500,000 tons of liquefied natural gas in southeast Asia for consumption in North America. Although in liquified form, the gas evaporates at a rate of 0.2 percent per day, implying $\rho = .998$.

5.8 THE OBJECTIVE FUNCTION

The final dimension of our model is the objective function. We divide this presentation from a summary of the contribution function (or cost function if we are minimizing) and the function we optimize to find the best policy.

5.8.1 The contribution function

We assume we have some measure of how well we are doing. This might be a cost that we wish to minimize, a contribution or reward if we are maximizing, or a more generic utility (which we typically maximize). We assume we are maximizing a contribution. In many problems, the contribution is a deterministic function of the state and action, in which case we would write

$C(S_t, a_t) =$ The contribution (cost if we are minimizing) earned by taking action a_t while in state S_t at time t.

We often write the contribution function as $C_t(S_t, a_t)$ to emphasize that it is being measured at time t and therefore depends on information in the state variable S_t. Our contribution may be random. For example, we may invest a dollars in an asset that earns a random return ρ_{t+1} which we do not know until time $t+1$. We may think of the contribution as $\rho_{t+1}a_t$, which means the contribution is random. In this case, we typically will write

$\hat{C}_{t+1}(S_t, a_t, W_{t+1}) =$ Contribution at time t from being in state S_t, making decision a_t which also depends on the information W_{t+1}.

We emphasize that the decision a_t does not have access to the information W_{t+1} (this is where our time indexing style eliminates any ambiguity about the information content of a variable). As a result, the decision a_t has to work with the expected contribution, which we write

$$C_t(S_t, a_t) = \mathbb{E}\{\hat{C}_{t+1}(S_t, a_t, W_{t+1})|S_t\}.$$

The role that W_{t+1} plays is problem-dependent, as illustrated in the examples below.

■ **EXAMPLE 5.1**

In asset acquisition problems, we order a_t in time period t to be used to satisfy demands $\hat{D}_{t+1}$ in the next time period. Our state variable is $S_t = R_t =$ the product on hand after demands in period t have been satisfied. We pay a cost $c^p a_t$ in period t and receive a revenue $p\min(R_t + a_t, \hat{D}_{t+1})$ in period $t+1$. Our total one-period contribution function is then

$$\hat{C}_{t,t+1}(R_t, a_t, \hat{D}_{t+1}) = p\min(R_t + a_t, \hat{D}_{t+1}) - c^p a_t.$$

The expected contribution is

$$C_t(S_t, a_t) = \mathbb{E}\{p\min(R_t + a_t, \hat{D}_{t+1}) - c^p a_t\}.$$

■ **EXAMPLE 5.2**

Now consider the same asset acquisition problem, but this time we place our orders in period t to satisfy the known demand in period t. Our cost function contains both a fixed cost c^f (which we pay for placing an order of any size) and a variable cost c^p. The cost function would look like

$$C_t(S_t, a_t) = \begin{cases} p\min(R_t + a_t, \hat{D}_t), & a_t = 0, \\ p\min(R_t + a_t, \hat{D}_t) - c^f - c^p a_t, & a_t > 0,. \end{cases}$$

Note that our contribution function no longer contains information from the next time period. If we did not incur a fixed cost c^f, then we would simply look at the demand D_t and order the quantity needed to cover demand (as a result, there would never be any product left over). However, since we incur a fixed cost c^f with each order, there is a benefit to ordering enough to cover the demand now and future demands. This benefit is captured through the value function.

There are many resource allocation problems where the contribution of a decision can be written using

c_{tra} = The unit contribution of acting on a resource with attribute vector r with action a.

This contribution is incurred in period t using information available in period t. In this case, our total contribution at time t could be written

$$C_t(S_t, a_t) = \sum_{r \in \mathcal{R}} \sum_{a \in \mathcal{A}} c_{tra} x_{tra}.$$

It is surprisingly common for us to want to work with two contributions. The common view of a contribution function is that it contains revenues and costs that we want to maximize or minimize. In many operational problems, there can be a mixture of “hard dollars” and “soft dollars.” The hard dollars are our quantifiable revenues and costs. But there are often other issues that are important in an operational setting, but which cannot always be easily quantified. For example, if we cannot cover all of the demand, we may wish to assess a penalty for not satisfying it. We can then manipulate this penalty to reduce the amount of unsatisfied demand. Examples of the use of soft-dollar bonuses and penalties abound in operational problems (see examples).

■ **EXAMPLE 5.1**

A trucking company has to pay the cost of a driver to move a load, but wants to avoid using inexperienced drivers for their high priority accounts (but has to accept the fact that it is sometimes necessary). An artificial penalty can be used to reduce the number of times this happens.

■ **EXAMPLE 5.2**

A charter jet company requires that in order for a pilot to land at night, he/she has to have landed a plane at night three times in the last 60 days. If the third time a pilot landed at night is at least 50 days ago, the company wants to encourage assignments of these pilots to flights with night landings so that they can maintain their status. A bonus can be assigned to encourage these assignments.

■ **EXAMPLE 5.3**

A student planning her schedule of courses has to face the possibility of failing a course, which may require taking either an extra course one semester or a summer course. She wants to plan out her course schedule as a dynamic program, but use a penalty to reduce the likelihood of having to take an additional course.

■ **EXAMPLE 5.4**

An investment banker wants to plan a strategy to maximize the value of an asset and minimize the likelihood of a very poor return. She is willing to accept lower overall returns in order to achieve this goal and can do it by incorporating an additional penalty when the asset is sold at a significant loss.

Given the presence of these so-called "soft dollars," it is useful to think of two contribution functions. We can let $C_t(S_t, a_t)$ be the hard dollars and $C_t^\pi(S_t, a_t)$ be the contribution function with the soft dollars included. The notation captures the fact that a set of soft bonuses and penalties represents a form of policy. So we can think of our policy as making decisions that maximize $C_t^\pi(S_t, a_t)$, but measure the value of the policy (in hard dollars), using $C_t(S_t, A^\pi(S_t))$.

5.8.2 Finding the best policy

We are now ready to optimize to find the best policy. Let $A_t^\pi(S_t)$ be a decision function (equivalent to a policy) that determines what decision we make given that we are in state S_t. Our optimization problem is to choose the best policy by choosing the best decision function from the family $(A_t^\pi(S_t))_{\pi\in\Pi}$. We wish to choose the best function that maximizes the total expected (discounted) contribution over a finite (or infinite) horizon. This would be written as

$$F_0^* = \max_{\pi\in\Pi} \mathbb{E}^\pi \left\{ \sum_{t=0}^{T} \gamma^t C_t^\pi(S_t, A_t^\pi(S_t)) | S_0 \right\}, \tag{5.16}$$

where γ discounts the money into time $t = 0$ values. We write the value of policy π as

$$F_0^\pi = \mathbb{E} \left\{ \sum_{t=0}^{T} \gamma^t C_t^\pi(S_t, A_t^\pi(S_t)) | S_0 \right\}.$$

In some communities, it is common to use an interest rate r, in which case the discount factor is

$$\gamma = \frac{1}{1+r}.$$

Important variants of this objective function are the infinite horizon problem ($T = \infty$), the undiscounted finite horizon problem ($\gamma = 1$), and the average reward, given by

$$F_0^\pi = \mathbb{E} \left\{ \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} C_t^\pi(S_t, A_t^\pi(S_t)) | S_0 \right\}. \tag{5.17}$$

A word of explanation is in order when we write the expectation 5.16 as $\mathbb{E}^\pi(\cdot)$. If we can pre-generate all the potential outcomes $W(\omega)$ in advance (without regard to the dynamics of the system), then we would normally write the expectation $\mathbb{E}(\cdot)$, since the exogenous events are not affected by the policy. Of course, the policy does affect the dynamics, but as long as it does not affect the random events, then the expectation does not depend on the policy. However, there are many problems where the exogenous events depend on the states that we visit, and possibly the actions we take. For this reason, it is safest to express the expectation as dependent on the policy.

Our optimization problem is to choose the best policy. In most practical applications, we can write the optimization problem as one of choosing the best policy, or

$$F_0^* = \max_{\pi \in \Pi} F_0^\pi. \tag{5.18}$$

It might be the case that a policy is characterized by a continuous parameter (the speed of a car, the temperature of a process, the price of an asset). In theory, we could have a problem where the optimal policy corresponds to a value of a parameter being equal to infinity. It is possible that F_0^* exists, but that an optimal "policy" does not exist (because it requires finding a parameter equal to infinity). While this is more of a mathematical curiosity, we handle these situations by writing the optimization problem as

$$F_0^* = \sup_{\pi \in \Pi} F_0^\pi, \tag{5.19}$$

where "sup" is the supremum operator, which finds the smallest number greater than or equal to F_0^π for any value of π. If we were minimizing, we would use "$\inf$," which stands for "infimum," which is the largest value less than or equal to the value of any policy. It is common in more formal treatments to use "$\sup$" instead of "$\max$" or "$\inf$" instead of "$\min$" since these are more general. Our emphasis is on computation and approximation, where we consider only problems where a solution exists. For this reason, we use "$\max$" and "$\min$" throughout our presentation.

The expression (5.16) contains one important but subtle assumption that will prove to be critical later and which will limit the applicability of our techniques in some problem classes. Specifically, we assume the presence of what is known as *linear, additive utility*. That is, we have added up contributions for each time period. It does not matter if the contributions are discounted or if the contribution functions themselves are nonlinear. However, we will not be able to handle functions that look like

$$F^\pi = \mathbb{E}^\pi \left\{ \left(\sum_{t \in \mathcal{T}} \gamma^t C_t(S_t, A_t^\pi(S_t)) \right)^2 \right\}. \tag{5.20}$$

The assumption of linear, additive utility means that the total contribution is a separable function of the contributions in each time period. While this works for many problems, it certainly does not work for all of them, as depicted in the examples below.

■ **EXAMPLE 5.1**

We may value a policy of managing a resource using a nonlinear function of the number of times the price of a resource dropped below a certain amount.

■ **EXAMPLE 5.2**

Assume we have to find the route through a network where the traveler is trying to arrive at a particular point in time. The value function is a nonlinear function of the total lateness, which means that the value function is not a separable function of the delay on each link.

■ **EXAMPLE 5.3**

Consider a mutual fund manager who has to decide how much to allocate between aggressive stocks, conservative stocks, bonds, and money market instruments. Let the allocation of assets among these alternatives represent a policy π. The mutual fund manager wants to maximize long term return, but needs to be sensitive to short term swings (the risk). He can absorb occasional downswings, but wants to avoid sustained downswings over several time periods. Thus, his value function must consider not only his return in a given time period, but also how his return looks over one-year, three-year and five-year periods.

In some cases these apparent instances of violations of linear, additive utility can be solved using a creatively defined state variable.

5.9 A MEASURE-THEORETIC VIEW OF INFORMATION**

For researchers 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 outcomes in Ω. An understanding of sigma-algebras is not important for computational work, but can be useful in certain types of proofs, as we see in the "why does it work" sections at the end of several chapters. 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.

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 $(\Omega, \mathfrak{F}, \mathcal{P})$. If we wish to take an expectation of some quantity that depends on the information, say $Ef(W_t)$, then we would sum (or integrate) over the set ω multiplied by the probability (or density) $\mathcal{P}$.

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 5.2.

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 5.2 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 Ω 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.

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).

5.10 BIBLIOGRAPHIC NOTES

Section 5.2 - Figure 5.1 which describes the mapping from continuous to discrete time was outlined for me by Erhan Cinlar.

Section 5.3 - The multiattribute notation for multiple resource classes is based primarily on Simao et al. (2001).

Section 5.4 - The definition of states is amazingly confused in the stochastic control literature. The first recognition of the difference between the physical state and the state of knowledge appears to be in Bellman & Kalaba (1959) which used the term "hyperstate" to refer to the state of knowledge. The control literature has long used state to represent a sufficient statistic (see for example Kirk (1998)), 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. See also Guestrin et al. (2003) for an application of the concept of factored state spaces to a Markov decision process.

Section 5.5 - 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 Simao 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 (1998) and Bertsekas (2005). For examples of treatments of dynamic programming in economics, see Stokey & R. E. Lucas (1989) and Chow (1997).

Section 5.6 - 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. An example includes Werbos (1992*b*).

PROBLEMS

5.1 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.

5.2 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.

5.3 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?

5.4 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.

5.5 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)?

5.6 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.

5.7 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.

5.8 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.

5.9 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).

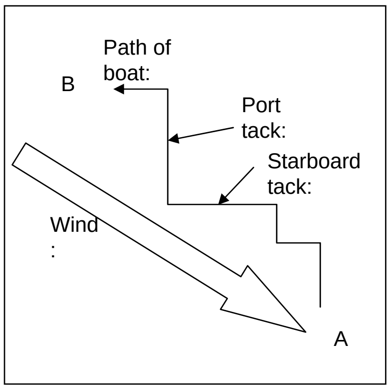
(b) Next assume that the cost c_{ij} is a random variable with an unknown distribution. Each time you traverse a link (i,j), you observe the cost $\hat{c}_{ij}$, which allows you to update your estimate $\bar{c}_{ij}$ of the mean of c_{ij}.

(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 $t = 1, 2, 3, \ldots$).

5.10 In the figure below, a sailboat is making its way upwind from point A to point B. To do this, the sailboat must tack, whereby it sails generally at a 45 degree angle to the wind. The problem is that the angle of the wind tends to shift randomly over time. The skipper decides to check the angle of the wind each minute and must decide whether the boat should be on port or starboard tack. Note that the proper decision must consider the current location of the boat, which we may indicate by an (x, y) coordinate.



(a) Formulate the problem as a dynamic program. Carefully define the state variable, decision variable, exogenous information and the contribution function.

(b) Use δ to discretize any continuous variables (in practice, you might choose different levels of discretization for each variable, but we are going to keep it simple). In terms of δ, give the size of the state space, the number of exogenous outcomes (in a single time period) and the action space. If you need an upper bound on a variable (e.g. wind speed), simply define an appropriate variable and express your answer in terms of this variable. All your answers should be expressed algebraically.

(c) Using a maximum wind speed of 30 miles per hour and $\delta = .1$, compute the size of your state, outcome and action spaces.

5.11 Implement your model from exercise 5.10 as a Markov decision process, and solve it using the techniques of 3 (section 3.2). Choose a value of δ that makes your program computationally reasonable (run times under 10 minutes). Let $\bar{\delta}$ be the smallest value of δ that produces a run time (for your computer) of under 10 minutes, and compare your solution (in terms of the total contribution) for $\delta = \bar{\delta}^N$ for $N = 2, 4, 8, 16$. Evaluate the quality of the solution by simulating 1000 iterations using the value functions obtained using backward dynamic programming. Plot your average contribution function as a function of δ.

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

5.13 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?

5.14 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.

5.15 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?

5.16 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.