'TRODUCTION TO OPERATIONS RESEARCH"
EDITION HILLIER + LIEBERMAN

CHAPTER 19

Markov Decision Processes

Chapter 16 introduced *Markov chains* and their analysis. Most of the chapter was devoted to *discrete time* Markov chains, i.e., Markov chains that are observed only at discrete points in time (e.g., the end of each day) rather than continuously. Each time it is observed, the Markov chain can be in any one of a number of *states*. Given the current state, a (one-step) *transition matrix* gives the probabilities for what the state will be next time. Given this transition matrix, Chap. 16 focused on *describing the behavior* of a Markov chain, e.g., finding the steady-state probabilities for what state it is in.

Many important systems (e.g., many queueing systems) can be modeled as either a discrete time or continuous time Markov chain. It is useful to describe the behavior of such a system (as we did in Chap. 17 for queueing systems) in order to evaluate its performance. However, it may be even more useful to *design the operation* of the system so as to *optimize its performance* (as we did in Sec. 17.10 for queueing systems).

This chapter focuses on how to design the operation of a discrete time Markov chain so as to optimize its performance. Therefore, rather than passively accepting the design of the Markov chain and the corresponding fixed transition matrix, we now are being proactive. For each possible state of the Markov chain, we make a decision about which one of several alternative actions should be taken in that state. The action chosen affects the *transition probabilities* as well as both the *immediate costs* (or rewards) and *subsequent costs* (or rewards) from operating the system. We want to choose the optimal actions for the respective states when considering both immediate and subsequent costs. The decision process for doing this is referred to as a *Markov decision process*.

The first section gives a prototype example of an application of a Markov decision process. Section 19.2 formulates the basic model for these processes. The next three sections describe how to solve them.

19.1 A PROTOTYPE EXAMPLE

A manufacturer has one key machine at the core of one of its production processes. Because of heavy use, the machine deteriorates rapidly in both quality and output. Therefore, at the end of each week, a thorough inspection is done that results in classifying the condition of the machine into one of four possible states:

State	Condition
0	Good as new
1	Operable—minor deterioration
2	Operable—major deterioration
3	Inoperable—output of unacceptable quality

After historical data on these inspection results are gathered, statistical analysis is done on how the state of the machine evolves from month to month. The following matrix shows the relative frequency (probability) of each possible transition from the state in one month (a row of the matrix) to the state in the following month (a column of the matrix).

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	0	$\frac{1}{2}$	$\frac{1}{2}$
3	0	0	0	1

In addition, statistical analysis has found that these transition probabilities are unaffected by also considering what the states were in prior months. This "lack-of-memory property" is the *Markovian property* described in Sec. 16.2. Therefore, for the random variable X_t, which is the state of the machine at the end of month t, it has been concluded that the stochastic process $\{X_t, t = 0, 1, 2, \ldots\}$ is a *discrete time Markov chain* whose (one-step) *transition matrix* is just the above matrix.

As the last entry in this transition matrix indicates, once the machine becomes inoperable (enters state 3), it remains inoperable. In other words, state 3 is an *absorbing state.* Leaving the machine in this state would be intolerable, since this would shut down the production process, so the machine must be replaced. (Repair is not feasible in this state.) The new machine then will start off in state 0.

The replacement process takes 1 week to complete so that production is lost for this period. The cost of the lost production (lost profit) is \$2,000, and the cost of replacing the machine is \$4,000, so the total cost incurred whenever the current machine enters state 3 is \$6,000.

Even before the machine reaches state 3, costs may be incurred from the production of defective items. The expected costs per week from this source are as follows:

State	Expected Cost Due to Defective Items, \$
0	0
1	1,000
2	3,000

We now have mentioned all the relevant costs associated with one particular *maintenance policy* (replace the machine when it becomes inoperable but do no maintenance otherwise). Under this policy, the evolution of the state of the *system* (the succession of machines) still is a Markov chain, but now with the following transition matrix:

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	0	$\frac{1}{2}$	$\frac{1}{2}$
3	1	0	0	0

To evaluate this maintenance policy, we should consider both the immediate costs incurred over the coming week (just described) and the subsequent costs that result from having the system evolve in this way. As introduced in Sec. 16.5, one such widely used measure of performance for Markov chains is the (long-run) **expected average cost per unit time.**[1]

To calculate this measure, we first derive the *steady-state probabilities* π_0, π_1, π_2, and π_3 for this Markov chain by solving the following steady-state equations:

$$\pi_0 = \pi_3,$$

$$\pi_1 = \frac{7}{8}\pi_0 + \frac{3}{4}\pi_1,$$

$$\pi_2 = \frac{1}{16}\pi_0 + \frac{1}{8}\pi_1 + \frac{1}{2}\pi_2,$$

$$\pi_3 = \frac{1}{16}\pi_0 + \frac{1}{8}\pi_1 + \frac{1}{2}\pi_2,$$

$$1 = \pi_0 + \pi_1 + \pi_2 + \pi_3.$$

(Although this system of equations is small enough to be solved by hand without great difficulty, the Steady-State Probabilities procedure in the Markov Chains area of your IOR Tutorial provides another quick way of obtaining this solution.) The simultaneous solution is

$$\pi_0 = \frac{2}{13}, \quad \pi_1 = \frac{7}{13}, \quad \pi_2 = \frac{2}{13}, \quad \pi_3 = \frac{2}{13}.$$

Hence, the (long-run) expected average cost per week for this maintenance policy is

$$0\pi_0 + 1{,}000\pi_1 + 3{,}000\pi_2 + 6{,}000\pi_3 = \frac{25{,}000}{13} = \$1{,}923.08.$$

However, there also are other maintenance policies that should be considered and compared with this one. For example, perhaps the machine should be replaced before it reaches state 3. Another alternative is to *overhaul* the machine at a cost of $2,000. This option is not feasible in state 3 and does not improve the machine while in state 0 or 1, so it is of interest only in state 2. In this state, an overhaul would return the machine to state 1. A week is required, so another consequence is $2,000 in lost profit from lost production.

In summary, the possible decisions after each inspection are as follows:

Decision	Action	Relevant States
1	Do nothing	0, 1, 2
2	Overhaul (return system to state 1)	2
3	Replace (return system to state 0)	1, 2, 3

[1]The term *long-run* indicates that the average should be interpreted as being taken over an *extremely* long time so that the effect of the initial state disappears. As time goes to infinity, Sec. 16.5 discusses the fact that the *actual* average cost per unit time essentially always converges to the *expected* average cost per unit time.

For easy reference, Table 19.1 also summarizes the relevant costs for each decision for each state where that decision could be of interest.

What is the optimal maintenance policy? We will be addressing this question to illustrate the material in the next four sections.

19.2 A MODEL FOR MARKOV DECISION PROCESSES

The model for the Markov decision processes considered in this chapter can be summarized as follows.

1. The state i of a discrete time Markov chain is observed after each transition ($i = 0, 1, \ldots, M$).
2. After each observation, a *decision* (action) k is chosen from a set of K possible decisions ($k = 1, 2, \ldots, K$). (Some of the K decisions may not be relevant for some of the states.)
3. If decision $d_i = k$ is made in state i, an immediate *cost* is incurred that has an expected value C_{ik}.
4. The decision $d_i = k$ in state i determines what the *transition probabilities*[1] will be for the next transition from state i. Denote these transition probabilities by $p_{ij}(k)$, for $j = 0, 1, \ldots, M$.
5. A specification of the decisions for the respective states ($d_0, d_1, \ldots, d_M$) prescribes a *policy* for the Markov decision process.
6. The objective is to find an *optimal policy* according to some cost criterion which considers both immediate costs and subsequent costs that result from the future evolution of the process. One common criterion is to minimize the (long-run) *expected average cost per unit time.* (An alternative criterion is considered in Sec. 19.5.)

To relate this general description to the prototype example presented in Sec. 19.1, recall that the Markov chain being observed there represents the state (condition) of a particular machine. After each inspection of the machine, a choice is made between three possible decisions (do nothing, overhaul, or replace). The resulting immediate expected cost is shown in the rightmost column of Table 19.1 for each relevant combination of state and decision. Section 19.1 analyzed one particular policy (d_0, d_1, d_2, d_3) = (1, 1, 1, 3), where decision 1 (do nothing) is made in states 0, 1, and 2 and decision 3 (replace) is made in state 3. The resulting transition probabilities are shown in the last transition matrix given in Sec. 19.1.

Our general model qualifies to be a *Markov* decision process because it possesses the Markovian property that characterizes any Markov process. In particular, given the current state and decision, any probabilistic statement about the future of the process is completely unaffected by providing any information about the history of the process. This

[1]The solution procedures given in the next two sections also assume that the resulting transition matrix is *irreducible.*

TABLE 19.1 Cost data for the prototype example

Decision	State	Expected Cost Due to Producing Defective Items, $	Maintenance Cost, $	Cost (Lost Profit) of Lost Production, $	Total Cost per Week, $
1. Do nothing	0	0	0	0	0
	1	1,000	0	0	1,000
	2	3,000	0	0	3,000
2. Overhaul	2	0	2,000	2,000	4,000
3. Replace	1, 2, 3	0	4,000	2,000	6,000

Markovian property holds here since (1) we are dealing with a Markov chain, (2) the new transition probabilities depend on only the current state and decision, and (3) the immediate expected cost also depends on only the current state and decision.

Our description of a policy implies two convenient (but unnecessary) properties that we will assume throughout the chapter (with one exception). One property is that a policy is **stationary;** i.e., whenever the system is in state i, the rule for making the decision always is the same regardless of the value of the current time t. The second property is that a policy is **deterministic;** i.e., whenever the system is in state i, the rule for making the decision definitely chooses one particular decision. (Because of the nature of the algorithm involved, the next section considers *randomized* policies instead, where a probability distribution is used for the decision to be made.)

Using this general framework, we now return to the prototype example and find the optimal policy by enumerating and comparing all the relevant policies. In doing this, we will let R denote a specific policy and $d_i(R)$ denote the corresponding decision to be made in state i, where decisions 1, 2, and 3 are described at the end of the preceding section. Since one or more of these three decisions are the only ones that would be considered in any given state, the only possible values of $d_i(R)$ are 1, 2, or 3 for any state i.

Solving the Prototype Example by Exhaustive Enumeration

The relevant policies for the prototype example are these:

Policy	Verbal Description	$d_0(R)$	$d_1(R)$	$d_2(R)$	$d_3(R)$
R_a	Replace in state 3	1	1	1	3
R_b	Replace in state 3, overhaul in state 2	1	1	2	3
R_c	Replace in states 2 and 3	1	1	3	3
R_d	Replace in states 1, 2, and 3	1	3	3	3

Each policy results in a different transition matrix, as shown below.

te	R_a			
	0	1	2	3
	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
	0	0	$\frac{1}{2}$	$\frac{1}{2}$
	1	0	0	0

State	R_b			
	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	1	0	0
3	1	0	0	0

te	R_c			
	0	1	2	3
	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
	1	0	0	0
	1	0	0	0

State	R_d			
	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0

From the rightmost column of Table 19.1, the values of C_{ik} are as follows:

State i \ Decision k	C_{ik} (in Thousands of Dollars)		
	1	2	3
0	0	—	—
1	1	—	6
2	3	4	6
3	—	—	6

As indicated in Sec. 16.5, the (long-run) expected average cost per unit time $E(C)$ then can be calculated from the expression

$$E(C) = \sum_{i=0}^{M} C_{ik}\pi_i,$$

where $k = d_i(R)$ for each i and $(\pi_0, \pi_1, \ldots, \pi_M)$ represents the steady-state distribution of the state of the system under the policy R being evaluated. After $(\pi_0, \pi_1, \ldots, \pi_M)$ are solved for under each of the four policies (as can be done with your IOR Tutorial), the calculation of $E(C)$ is as summarized here:

Policy	$(\pi_0, \pi_1, \pi_2, \pi_3)$	$E(C)$, in Thousands of Dollars
R_a	$\left(\frac{2}{13}, \frac{7}{13}, \frac{2}{13}, \frac{2}{13}\right)$	$\frac{1}{13}[2(0) + 7(1) + 2(3) + 2(6)] = \frac{25}{13} = \$1{,}923$
R_b	$\left(\frac{2}{21}, \frac{5}{7}, \frac{2}{21}, \frac{2}{21}\right)$	$\frac{1}{21}[2(0) + 15(1) + 2(4) + 2(6)] = \frac{35}{21} = \$1{,}667 \leftarrow$ Minimum
R_c	$\left(\frac{2}{11}, \frac{7}{11}, \frac{1}{11}, \frac{1}{11}\right)$	$\frac{1}{11}[2(0) + 7(1) + 1(6) + 1(6)] = \frac{19}{11} = \$1{,}727$
R_d	$\left(\frac{1}{2}, \frac{7}{16}, \frac{1}{32}, \frac{1}{32}\right)$	$\frac{1}{32}[16(0) + 14(6) + 1(6) + 1(6)] = \frac{96}{32} = \$3{,}000$

Thus, the optimal policy is R_b; that is, replace the machine when it is found to be in state 3, and overhaul the machine when it is found to be in state 2. The resulting (long-run) expected average cost per week is \$1,667.

If you would like to go through another small example, one is provided in the Worked Examples section of the CD-ROM.

Using exhaustive enumeration to find the optimal policy is appropriate for such tiny examples, where there are so few relevant policies. However, many applications have so many policies that this approach would be completely infeasible. For such cases, algorithms that can efficiently find an optimal policy are needed. The next three sections consider such algorithms.

19.3 LINEAR PROGRAMMING AND OPTIMAL POLICIES

Section 19.2 described the main kind of policy (called a *stationary, deterministic* policy) that is used by Markov decision processes. We saw that any such policy R can be viewed as a rule that prescribes decision $d_i(R)$ whenever the system is in state i, for each $i = 0, 1, \ldots, M$. Thus, R is characterized by the values

$$\{d_0(R), d_1(R), \ldots, d_M(R)\}.$$

Equivalently, R can be characterized by assigning values $D_{ik} = 0$ or 1 in the matrix

$$\text{State } i \begin{matrix} \\ 0 \\ 1 \\ \vdots \\ M \end{matrix} \overset{\text{Decision } k}{\begin{matrix} 1 & 2 & \cdots & K \end{matrix}} \begin{bmatrix} D_{01} & D_{02} & \cdots & D_{0K} \\ D_{11} & D_{12} & \cdots & D_{1K} \\ \cdots & \cdots & \cdots & \cdots \\ D_{M1} & D_{M2} & \cdots & D_{MK} \end{bmatrix},$$

where each D_{ik} ($i = 0, 1, \ldots, M$ and $k = 1, 2, \ldots, K$) is defined as

$$D_{ik} = \begin{cases} 1 & \text{if decision } k \text{ is to be made in state } i \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, each row in the matrix must contain a single 1 with the rest of the elements 0s. For example, the optimal policy R_b for the prototype example is characterized by the matrix

$$\text{State } i \begin{matrix} \\ 0 \\ 1 \\ 2 \\ 3 \end{matrix} \overset{\text{Decision } k}{\begin{matrix} 1 & 2 & 3 \end{matrix}} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

i.e., do nothing (decision 1) when the machine is in state 0 or 1, overhaul (decision 2) in state 2, and replace the machine (decision 3) when it is in state 3.

Randomized Policies

Introducing D_{ik} provides motivation for a *linear programming formulation.* It is hoped that the expected cost of a policy can be expressed as a linear function of D_{ik} or a related variable, subject to linear constraints. Unfortunately, the D_{ik} values are integers (0 or 1), and continuous variables are required for a linear programming formulation. This requirement can be handled by expanding the interpretation of a policy. The previous definition calls for making the same decision every time the system is in state i. The new interpretation of a policy will call for determining a probability distribution for the decision to be made when the system is in state i.

With this new interpretation, the D_{ik} now need to be redefined as

$$D_{ik} = P\{\text{decision} = k \mid \text{state} = i\}.$$

In other words, given that the system is in state i, variable D_{ik} is the *probability* of choosing decision k as the decision to be made. Therefore, $(D_{i1}, D_{i2}, \ldots, D_{iK})$ is the *probability distribution* for the decision to be made in state i.

This kind of policy using probability distributions is called a **randomized policy,** whereas the policy calling for $D_{ik} = 0$ or 1 is a *deterministic policy.* Randomized policies can again be characterized by the matrix

$$\text{State } i \begin{matrix} \\ 0 \\ 1 \\ \vdots \\ M \end{matrix} \overset{\text{Decision } k}{\begin{matrix} 1 & 2 & \cdots & K \end{matrix}} \begin{bmatrix} D_{01} & D_{02} & \cdots & D_{0K} \\ D_{11} & D_{12} & \cdots & D_{1K} \\ \cdots & \cdots & \cdots & \cdots \\ D_{M1} & D_{M2} & \cdots & D_{MK} \end{bmatrix},$$

where each row sums to 1, and now

$$0 \leq D_{ik} \leq 1.$$

To illustrate, consider a randomized policy for the prototype example given by the matrix

$$\begin{array}{c} \\ \text{State } i \end{array} \begin{array}{c} \text{Decision } k \\ \begin{array}{cccc} & 1 & 2 & 3 \\ 0 & 1 & 0 & 0 \\ 1 & \frac{1}{2} & 0 & \frac{1}{2} \\ 2 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ 3 & 0 & 0 & 1 \end{array} \end{array}.$$

This policy calls for *always* making decision 1 (do nothing) when the machine is in state 0. If it is found to be in state 1, it is left as is with probability $\frac{1}{2}$ and replaced with probability $\frac{1}{2}$, so a coin can be flipped to make the choice. If it is found to be in state 2, it is left as is with probability $\frac{1}{4}$, overhauled with probability $\frac{1}{4}$, and replaced with probability $\frac{1}{2}$. Presumably, a random device with these probabilities (possibly a table of random numbers) can be used to make the actual decision. Finally, if the machine is found to be in state 3, it always is overhauled.

By allowing randomized policies, so that the D_{ik} are continuous variables instead of integer variables, it now is possible to formulate a linear programming model for finding an optimal policy.

A Linear Programming Formulation

The convenient decision variables (denoted here by y_{ik}) for a linear programming model are defined as follows. For each $i = 0, 1, \ldots, M$ and $k = 1, 2, \ldots, K$, let y_{ik} be the steady-state unconditional probability that the system is in state i *and* decision k is made; i.e.,

$$y_{ik} = P\{\text{state} = i \text{ and decision} = k\}.$$

Each y_{ik} is closely related to the corresponding D_{ik} since, from the rules of conditional probability,

$$y_{ik} = \pi_i D_{ik},$$

where π_i is the steady-state probability that the Markov chain is in state i. Furthermore,

$$\pi_i = \sum_{k=1}^{K} y_{ik},$$

so that

$$D_{ik} = \frac{y_{ik}}{\pi_i} = \frac{y_{ik}}{\sum_{k=1}^{K} y_{ik}}.$$

There exist three sets of constraints on y_{ik}:

1. $\sum_{i=1}^{M} \pi_i = 1$ so that $\sum_{i=0}^{M} \sum_{k=1}^{K} y_{ik} = 1.$

2. From results on steady-state probabilities (see Sec. 16.5),[1]

$$\pi_j = \sum_{i=0}^{M} \pi_i p_{ij}$$

[1]The argument k is introduced in $p_{ij}(k)$ to indicate that the appropriate transition probability depends upon the decision k.

so that

$$\sum_{k=1}^{K} y_{jk} = \sum_{i=0}^{M} \sum_{k=1}^{K} y_{ik} p_{ij}(k), \qquad \text{for } j = 0, 1, \ldots, M.$$

3. $y_{ik} \geq 0, \qquad$ for $i = 0, 1, \ldots, M$ and $k = 1, 2, \ldots, K$.

The long-run expected average cost per unit time is given by

$$E(C) = \sum_{i=0}^{M} \sum_{k=1}^{K} \pi_i C_{ik} D_{ik} = \sum_{i=0}^{M} \sum_{k=1}^{K} C_{ik} y_{ik}.$$

Hence, the linear programming model is to choose the y_{ik} so as to

$$\text{Minimize} \qquad Z = \sum_{i=0}^{M} \sum_{k=1}^{K} C_{ik} y_{ik},$$

subject to the constraints

$$(1) \qquad \sum_{i=0}^{M} \sum_{k=1}^{K} y_{ik} = 1.$$

$$(2) \qquad \sum_{k=1}^{K} y_{jk} - \sum_{i=0}^{M} \sum_{k=1}^{K} y_{ik} p_{ij}(k) = 0, \qquad \text{for } j = 0, 1, \ldots, M.$$

$$(3) \qquad y_{ik} \geq 0, \qquad \text{for } i = 0, 1, \ldots, M; \; k = 1, 2, \ldots, K.$$

Thus, this model has $M + 2$ functional constraints and $K(M + 1)$ decision variables. [Actually, (2) provides one *redundant* constraint, so any one of these $M + 1$ constraints can be deleted.]

Because this is a linear programming model, it can be solved by the *simplex method.* Once the y_{ik} values are obtained, each D_{ik} is found from

$$D_{ik} = \frac{y_{ik}}{\sum_{k=1}^{K} y_{ik}}.$$

The optimal solution obtained by the simplex method has some interesting properties. It will contain $M + 1$ basic variables $y_{ik} \geq 0$. It can be shown that $y_{ik} > 0$ for at least one $k = 1, 2, \ldots, K$, for each $i = 0, 1, \ldots, M$. Therefore, it follows that $y_{ik} > 0$ for only *one* k for each $i = 0, 1, \ldots, M$. Consequently, each $D_{ik} = 0$ or 1.

The key conclusion is that the optimal policy found by the simplex method is *deterministic* rather than randomized. Thus, allowing policies to be randomized does not help at all in improving the final policy. However, it serves an extremely useful role in this formulation by converting integer variables (the D_{ik}) to continuous variables so that linear programming (LP) can be used. (The analogy in *integer programming* is to use the *LP relaxation* so that the simplex method can be applied and then to have the *integer solutions property* hold so that the optimal solution for the LP relaxation turns out to be integer anyway.)

Solving the Prototype Example by Linear Programming

Refer to the prototype example of Sec. 19.1. The first two columns of Table 19.1 give the relevant combinations of states and decisions. Therefore, the decision variables that need to be included in the model are y_{01}, y_{11}, y_{13}, y_{21}, y_{22}, y_{23}, and y_{33}. (The general expressions given above for the model include y_{ik} for *irrelevant* combinations of states and

decisions here, so these $y_{ik} = 0$ in an optimal solution, and they might as well be deleted at the outset.) The rightmost column of Table 19.1 provides the coefficients of these variables in the objective function. The transition probabilities $p_{ij}(k)$ for each relevant combination of state i and decision k also are spelled out in Sec. 19.1.

The resulting linear programming model is

$$\text{Minimize} \quad Z = 1{,}000y_{11} + 6{,}000y_{13} + 3{,}000y_{21} + 4{,}000y_{22} + 6{,}000y_{23} + 6{,}000y_{33},$$

subject to

$$\begin{aligned} y_{01} + y_{11} + y_{13} + y_{21} + y_{22} + y_{23} + y_{33} &= 1 \\ y_{01} - (y_{13} + y_{23} + y_{33}) &= 0 \\ y_{11} + y_{13} - \left(\frac{7}{8}y_{01} + \frac{3}{4}y_{11} + y_{22}\right) &= 0 \\ y_{21} + y_{22} + y_{23} - \left(\frac{1}{16}y_{01} + \frac{1}{8}y_{11} + \frac{1}{2}y_{21}\right) &= 0 \\ y_{33} - \left(\frac{1}{16}y_{01} + \frac{1}{8}y_{11} + \frac{1}{2}y_{21}\right) &= 0 \end{aligned}$$

and

$$\text{all } y_{ik} \geq 0.$$

Applying the simplex method, we obtain the optimal solution

$$y_{01} = \frac{2}{21}, \quad (y_{11}, y_{13}) = \left(\frac{5}{7}, 0\right), \quad (y_{21}, y_{22}, y_{23}) = \left(0, \frac{2}{21}, 0\right), \quad y_{33} = \frac{2}{21},$$

so

$$D_{01} = 1, \quad (D_{11}, D_{13}) = (1, 0), \quad (D_{21}, D_{22}, D_{23}) = (0, 1, 0), \quad D_{33} = 1.$$

This policy calls for leaving the machine as is (decision 1) when it is in state 0 or 1, overhauling it (decision 2) when it is in state 2, and replacing it (decision 3) when it is in state 3. This is the same optimal policy found by exhaustive enumeration at the end of Sec. 19.2.

The Worked Examples section of the CD-ROM provides another example of applying linear programming to obtain an optimal policy for a Markov decision process.

19.4 POLICY IMPROVEMENT ALGORITHM FOR FINDING OPTIMAL POLICIES

You now have seen two methods for deriving an optimal policy for a Markov decision process: *exhaustive enumeration* and *linear programming*. Exhaustive enumeration is useful because it is both quick and straightforward for very small problems. Linear programming can be used to solve vastly larger problems, and software packages for the simplex method are very widely available.

We now present a third popular method, namely, a *policy improvement algorithm*. The key advantage of this method is that it tends to be very efficient, because it usually reaches an optimal policy in a relatively small number of iterations (far fewer than for the simplex method with a linear programming formulation).

By following the model of Sec. 19.2 and as a joint result of the current state i of the system and the decision $d_i(R) = k$ when operating under policy R, two things occur. An (expected) cost C_{ik} is incurred that depends upon only the observed state of the system and the decision made. The system moves to state j at the next observed time period, with

transition probability given by $p_{ij}(k)$. If, in fact, state j influences the cost that has been incurred, then C_{ik} is calculated as follows. Let

$q_{ij}(k)$ = expected cost incurred when the system is in state i, decision k is made, and the system evolves to state j at the next observed time period.

Then

$$C_{ik} = \sum_{j=0}^{M} q_{ij}(k)p_{ij}(k).$$

Preliminaries

Referring to the description and notation for Markov decision processes given at the beginning of Sec. 19.2, we can show that, for any given policy R, there exist values $g(R)$, $v_0(R)$, $v_1(R)$, . . . , $v_M(R)$ that satisfy

$$g(R) + v_i(R) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R), \qquad \text{for } i = 0, 1, 2, \ldots, M.$$

We now shall give a heuristic justification of these relationships and an interpretation for these values.

Denote by $v_i^n(R)$ the total expected cost of a system starting in state i (beginning the first observed time period) and evolving for n time periods. Then $v_i^n(R)$ has two components: C_{ik}, the cost incurred during the first observed time period, and $\sum_{j=0}^{M} p_{ij}(k)\, v_j^{n-1}(R)$, the total expected cost of the system evolving over the remaining $n - 1$ time periods. This gives the *recursive equation*

$$v_i^n(R) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j^{n-1}(R), \qquad \text{for } i = 0, 1, 2, \ldots, M,$$

where $v_i^1(R) = C_{ik}$ for all i.

It will be useful to explore the behavior of $v_i^n(R)$ as n grows large. Recall that the (long-run) expected average cost per unit time following any policy R can be expressed as

$$g(R) = \sum_{i=0}^{M} \pi_i C_{ik},$$

which is independent of the starting state i. Hence, $v_i^n(R)$ behaves approximately as $n\, g(R)$ for large n. In fact, if we neglect small fluctuations, $v_i^n(R)$ can be expressed as the sum of two components

$$v_i^n(R) \approx n\, g(R) + v_i(R),$$

where the first component is independent of the initial state and the second is dependent upon the initial state. Thus, $v_i(R)$ can be interpreted as the effect on the total expected cost due to starting in state i. Consequently,

$$v_i^n(R) - v_j^n(R) \approx v_i(R) - v_j(R),$$

so that $v_i(R) - v_j(R)$ is a measure of the effect of starting in state i rather than state j.

Letting n grow large, we now can substitute $v_i^n(R) = n\, g(R) + v_i(R)$ and $v_j^{n-1}(R) = (n - 1)g(R) + v_j(R)$ into the *recursive equation*. This leads to the system of equations given in the opening paragraph of this subsection.

Note that this system has $M + 1$ equations with $M + 2$ unknowns, so that one of these variables may be chosen arbitrarily. By convention, $v_M(R)$ will be chosen equal to zero.

Therefore, by solving the system of linear equations, we can obtain $g(R)$, the (long-run) expected average cost per unit time when policy R is followed. In principle, all policies can be enumerated and that policy which minimizes $g(R)$ can be found. However, even for a moderate number of states and decisions, this technique is cumbersome. Fortunately, there exists an algorithm that can be used to evaluate policies and find the optimal one without complete enumeration, as described next.

The Policy Improvement Algorithm

The algorithm begins by choosing an arbitrary policy R_1. It then solves the system of equations to find the values of $g(R_1)$, $v_0(R)$, $v_1(R)$, . . . , $v_{M-1}(R)$ [with $v_M(R) = 0$]. This step is called *value determination.* A better policy, denoted by R_2, is then constructed. This step is called *policy improvement.* These two steps constitute an iteration of the algorithm. Using the new policy R_2, we perform another iteration. These iterations continue until two successive iterations lead to identical policies, which signifies that the optimal policy has been obtained. The details are outlined below.

Summary of the Policy Improvement Algorithm

Initialization: Choose an arbitrary initial trial policy R_1. Set $n = 1$.
Iteration n:
Step 1: Value determination: For policy R_n, use $p_{ij}(k)$, C_{ik}, and $v_M(R_n) = 0$ to solve the system of $M + 1$ equations

$$g(R_n) = C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n) - v_i(R_n), \qquad \text{for } i = 0, 1, \ldots, M,$$

for all $M + 1$ unknown values of $g(R_n)$, $v_0(R_n)$, $v_1(R_n)$, . . . , $v_{M-1}(R_n)$.
Step 2: Policy improvement: Using the current values of $v_i(R_n)$ computed for policy R_n, find the alternative policy R_{n+1} such that, for each state i, $d_i(R_{n+1}) = k$ is the decision that minimizes

$$C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n) - v_i(R_n)$$

i.e., for *each* state i,

$$\underset{k=1, 2, \ldots, k}{\text{Minimize}} \qquad [C_{ik} + \sum_{j=0}^{M} p_{ij}(k)\, v_j(R_n)' - v_i(R_n)],$$

and then set $d_i(R_{n+1})$ equal to the minimizing value of k. This procedure defines a new policy R_{n+1}.
Optimality test: The current policy R_{n+1} is optimal if this policy is identical to policy R_n. If it is, stop. Otherwise, reset $n = n + 1$ and perform another iteration.

Two key properties of this algorithm are

1. $g(R_{n+1}) \leq g(R_n)$, for $n = 1, 2, \ldots$.
2. The algorithm terminates with an optimal policy in a finite number of iterations.[1]

[1]This termination is guaranteed under the assumptions of the model given in Sec. 19.2, including particularly the (implicit) assumptions of a finite number of states ($M + 1$) and a finite number of decisions (K), but not necessarily for more general models. See R. Howard, *Dynamic Programming and Markov Processes,* M.I.T. Press, Cambridge, MA, 1960. Also see pp. 1291–1293 in A. F. Veinott, Jr., "On Finding Optimal Policies in Discrete Dynamic Programming with No Discounting," *Annals of Mathematical Statistics,* **37:** 1284–1294, 1966.

Solving the Prototype Example by the Policy Improvement Algorithm

Referring to the prototype example presented in Sec. 19.1, we outline the application of the algorithm next.

Initialization. For the initial trial policy R_1, we arbitrarily choose the policy that calls for replacement of the machine (decision 3) when it is found to be in state 3, but doing nothing (decision 1) in other states. This policy, its transition matrix, and its costs are summarized below.

Policy R_1

State	Decision
0	1
1	1
2	1
3	3

Transition matrix

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	0	$\frac{1}{2}$	$\frac{1}{2}$
3	1	0	0	0

Costs

State	C_{ik}
0	0
1	1,000
2	3,000
3	6,000

Iteration 1. With this policy, the value determination step requires solving the following four equations simultaneously for $g(R_1)$, $v_0(R_1)$, $v_1(R_1)$, and $v_2(R_1)$ [with $v_3(R_1) = 0$].

$$g(R_1) = + \frac{7}{8}v_1(R_1) + \frac{1}{16}v_2(R_1) - v_0(R_1).$$

$$g(R_1) = 1{,}000 + \frac{3}{4}v_1(R_1) + \frac{1}{8}v_2(R_1) - v_1(R_1).$$

$$g(R_1) = 3{,}000 + \frac{1}{2}v_2(R_1) - v_2(R_1).$$

$$g(R_1) = 6{,}000 + v_0(R_1).$$

The simultaneous solution is

$$g(R_1) = \frac{25{,}000}{13} = 1{,}923$$

$$v_0(R_1) = -\frac{53{,}000}{13} = -4{,}077$$

$$v_1(R_1) = -\frac{34{,}000}{13} = -2{,}615$$

$$v_2(R_1) = \frac{28{,}000}{13} = 2{,}154.$$

Step 2 (policy improvement) can now be applied. We want to find an improved policy R_2 such that decision k in state i minimizes the corresponding expression below.

State 0: $C_{0k} - p_{00}(k)(4{,}077) - p_{01}(k)(2{,}615) + p_{02}(k)(2{,}154) + 4{,}077$
State 1: $C_{1k} - p_{10}(k)(4{,}077) - p_{11}(k)(2{,}615) + p_{12}(k)(2{,}154) + 2{,}615$
State 2: $C_{2k} - p_{20}(k)(4{,}077) - p_{21}(k)(2{,}615) + p_{22}(k)(2{,}154) - 2{,}154$
State 3: $C_{3k} - p_{30}(k)(4{,}077) - p_{31}(k)(2{,}615) + p_{32}(k)(2{,}154).$

Actually, in state 0, the only decision allowed is decision 1 (do nothing), so no calculations are needed. Similarly, we know that decision 3 (replace) must be made in state 3.

Thus, only states 1 and 2 require calculation of the values of these expressions for alternative decisions.

For state 1, the possible decisions are 1 and 3. For each one, we show below the corresponding C_{1k}, the $p_{1j}(k)$, and the resulting value of the expression.

	State 1						
Decision	C_{1k}	$p_{10}(k)$	$p_{11}(k)$	$p_{12}(k)$	$p_{13}(k)$	**Value of Expression**	
1	1,000	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	1,923	← Minimum
3	6,000	1	0	0	0	4,538	

Since decision 1 minimizes the expression, it is chosen as the decision to be made in state 1 for policy R_2 (just as for policy R_1).

The corresponding results for state 2 are shown below for its three possible decisions.

	State 2						
Decision	C_{2k}	$p_{20}(k)$	$p_{21}(k)$	$p_{22}(k)$	$p_{23}(k)$	**Value of Expression**	
1	3,000	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1,923	
2	4,000	0	1	0	0	−769	← Minimum
3	6,000	1	0	0	0	−231	

Therefore, decision 2 is chosen as the decision to be made in state 2 for policy R_2. Note that this is a change from policy R_1.

We summarize our new policy, its transition matrix, and its costs below.

Policy R_2

State	**Decision**
0	1
1	1
2	2
3	3

Transition matrix

State	**0**	**1**	**2**	**3**
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	1	0	0
3	1	0	0	0

Costs

State	C_{ik}
0	0
1	1,000
2	4,000
3	6,000

Since this policy is not identical to policy R_1, the optimality test says to perform another iteration.

Iteration 2. For step 1 (value determination), the equations to be solved for this policy are shown below.

$$g(R_2) = \quad + \frac{7}{8}v_1(R_2) + \frac{1}{16}v_2(R_2) - v_0(R_2).$$

$$g(R_2) = 1{,}000 \quad + \frac{3}{4}v_1(R_2) + \frac{1}{8}v_2(R_2) - v_1(R_2).$$

$$g(R_2) = 4{,}000 \quad + v_1(R_2) \quad - v_2(R_2).$$

$$g(R_2) = 6{,}000 + v_0(R_2).$$

The simultaneous solution is

$$g(R_2) = \frac{5{,}000}{3} = 1{,}667$$

$$v_0(R_2) = -\frac{13{,}000}{3} = -4{,}333$$

$$v_1(R_2) = -3{,}000$$

$$v_2(R_2) = -\frac{2{,}000}{3} = -667.$$

Step 2 (policy improvement) can now be applied. For the two states with more than one possible decision, the expressions to be minimized are

State 1: $C_{1k} - p_{10}(k)(4{,}333) - p_{11}(k)(3{,}000) - p_{12}(k)(667) + 3{,}000$

State 2: $C_{2k} - p_{20}(k)(4{,}333) - p_{21}(k)(3{,}000) - p_{22}(k)(667) + 667.$

The first iteration provides the necessary data (the transition probabilities and C_{ik}) required for determining the new policy, except for the values of each of these expressions for each of the possible decisions. These values are

Decision	Value for State 1	Value for State 2
1	1,667	3,333
2	—	1,667
3	4,667	2,334

Since decision 1 minimizes the expression for state 1 and decision 2 minimizes the expression for state 2, our next trial policy R_3 is

Policy R_3

State	Decision
0	1
1	1
2	2
3	3

Note that policy R_3 is identical to policy R_2. Therefore, the optimality test indicates that this policy is optimal, so the algorithm is finished.

Another example illustrating the application of this algorithm is included in your OR Tutor. The Worked Examples section of the CD-ROM provides an additional example as well. The IOR Tutorial also includes an *interactive* procedure for efficiently learning and applying the algorithm.

19.5 DISCOUNTED COST CRITERION

Throughout this chapter, we have measured policies on the basis of their (long-run) expected average cost per unit time. We now turn to an alternative measure of performance, namely, the **expected total discounted cost.**

As first introduced in Sec. 18.2, this measure uses a *discount factor* α, where $0 < \alpha < 1$. The discount factor α can be interpreted as equal to $1/(1 + i)$, where i is the

current interest rate per period. Thus, α is the *present value* of one unit of cost one period in the future. Similarly, α^m is the *present value* of one unit of cost m periods in the future.

This *discounted cost criterion* becomes preferable to the *average cost criterion* when the time periods for the Markov chain are sufficiently long that the *time value of money* should be taken into account in adding costs in future periods to the cost in the current period. Another advantage is that the discounted cost criterion can readily be adapted to dealing with a *finite-period* Markov decision process where the Markov chain will terminate after a certain number of periods.

Both the policy improvement technique and the linear programming approach still can be applied here with relatively minor adjustments from the average cost case, as we describe next. Then we will present another technique, called the *method of successive approximations,* for quickly approximating an optimal policy.

A Policy Improvement Algorithm

To derive the expressions needed for the value determination and policy improvement steps of the algorithm, we now adopt the viewpoint of *probabilistic dynamic programming* (as described in Sec. 10.4). In particular, for each state i ($i = 0, 1, \ldots, M$) of a Markov decision process operating under policy R, let $V_i^n(R)$ be the *expected total discounted cost* when the process starts in state i (beginning the first observed time period) and evolves for n time periods. Then $V_i^n(R)$ has two components: C_{ik}, the cost incurred during the first observed time period, and $\alpha \sum_{j=0}^{M} p_{ij}(k)V_j^{n-1}(R)$, the expected total discounted cost of the process evolving over the remaining $n - 1$ time periods. For each $i = 0, 1, \ldots, M$, this yields the recursive equation

$$V_i^n(R) = C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k)V_j^{n-1}(R),$$

with $V_i^1(R) = C_{ik}$, which closely resembles the recursive relationships of probabilistic dynamic programming found in Sec. 10.4.

As n approaches infinity, this recursive equation converges to

$$V_i(R) = C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k)V_j(R), \qquad \text{for } i = 0, 1, \ldots, M,$$

where $V_i(R)$ can now be interpreted as the expected total discounted cost when the process starts in state i and continues indefinitely. There are $M + 1$ equations and $M + 1$ unknowns, so the simultaneous solution of this system of equations yields the $V_i(R)$.

To illustrate, consider again the prototype example of Sec. 19.1. Under the average cost criterion, we found in Secs. 19.2, 19.3, and 19.4 that the optimal policy is to do nothing in states 0 and 1, overhaul in state 2, and replace in state 3. Under the discounted cost criterion, with $\alpha = 0.9$, this same policy gives the following system of equations:

$$V_0(R) = \quad + 0.9\left[\quad \frac{7}{8}V_1(R) + \frac{1}{16}V_2(R) + \frac{1}{16}V_3(R)\right]$$

$$V_1(R) = 1{,}000 + 0.9\left[\quad \frac{3}{4}V_1(R) + \frac{1}{8}V_2(R) + \frac{1}{8}V_3(R)\right]$$

$$V_2(R) = 4{,}000 + 0.9[\quad V_1(R)]$$

$$V_3(R) = 6{,}000 + 0.9[V_0(R)].$$

The simultaneous solution is

$$V_0(R) = 14{,}949$$
$$V_1(R) = 16{,}262$$
$$V_2(R) = 18{,}636$$
$$V_3(R) = 19{,}454.$$

Thus, assuming that the system starts in state 0, the expected total discounted cost is \$14,949.

This system of equations provides the expressions needed for a policy improvement algorithm. After summarizing this algorithm in general terms, we shall use it to check whether this particular policy still is optimal under the discounted cost criterion.

Summary of the Policy Improvement Algorithm (Discounted Cost Criterion)

Initialization: Choose an arbitrary initial trial policy R_1. Set $n = 1$.

Iteration n:

Step 1: Value determination: For policy R_n, use $p_{ij}(k)$ and C_{ik} to solve the system of $M + 1$ equations

$$V_i(R_n) = C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k) V_j(R_n), \qquad \text{for } i = 0, 1, \ldots, M,$$

for all $M + 1$ unknown values of $V_0(R_n), V_1(R_n), \ldots, V_M(R_n)$.

Step 2: Policy improvement: Using the current values of the $V_i(R_n)$, find the alternative policy R_{n+1} such that, for each state i, $d_i(R_{n+1}) = k$ is the decision that minimizes

$$C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k) V_j(R_n)$$

i.e., for *each* state i,

$$\underset{k=1, 2, \ldots, K}{\text{Minimize}} \left[C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k) V_j(R_n) \right],$$

and then set $d_i(R_{n+1})$ equal to the minimizing value of k. This procedure defines a new policy R_{n+1}.

Optimality test: The current policy R_{n+1} is optimal if this policy is identical to policy R_n. If it is, stop. Otherwise, reset $n = n + 1$ and perform another iteration.

Three key properties of this algorithm are as follows:

1. $V_i(R_{n+1}) \leq V_i(R_n)$, for $i = 0, 1, \ldots, M$ and $n = 1, 2, \ldots$.
2. The algorithm terminates with an optimal policy in a finite number of iterations.
3. The algorithm is valid without the assumption (used for the average cost case) that the Markov chain associated with every transition matrix is irreducible.

Your IOR Tutorial includes an *interactive* procedure for applying this algorithm.

Solving the Prototype Example by This Policy Improvement Algorithm. We now pick up the prototype example where we left it before summarizing the algorithm.

We already have selected the optimal policy under the average cost criterion to be our initial trial policy R_1. This policy, its transition matrix, and its costs are summarized below.

Policy R_1

State	Decision
0	1
1	1
2	2
3	3

Transition matrix

State	0	1	2	3
0	0	$\frac{7}{8}$	$\frac{1}{16}$	$\frac{1}{16}$
1	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
2	0	1	0	0
3	1	0	0	0

Costs

State	C_{ik}
0	0
1	1,000
2	4,000
3	6,000

We also have already done step 1 (value determination) of iteration 1. This transition matrix and these costs led to the system of equations used to find $V_0(R_1) = 14{,}949$, $V_1(R_1) = 16{,}262$, $V_2(R_1) = 18{,}636$, and $V_3(R_1) = 19{,}454$.

To start step 2 (policy improvement), we only need to construct the expression to be minimized for the two states (1 and 2) with a choice of decisions.

State 1: $$C_{1k} + 0.9[p_{10}(k)(14{,}949) + p_{11}(k)(16{,}262) + p_{12}(k)(18{,}636) + p_{13}(k)(19{,}454)]$$

State 2: $$C_{2k} + 0.9[p_{20}(k)(14{,}949) + p_{21}(k)(16{,}262) + p_{22}(k)(18{,}636) + p_{23}(k)(19{,}454)].$$

For each of these states and their possible decisions, we show below the corresponding C_{ik}, the $p_{ij}(k)$, and the resulting value of the expression.

	State 1						
Decision	C_{1k}	$p_{10}(k)$	$p_{11}(k)$	$p_{12}(k)$	$p_{13}(k)$	Value of Expression	
1	1,000	0	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$	16,262	← Minimum
3	6,000	1	0	0	0	19,454	

	State 2						
Decision	C_{2k}	$p_{20}(k)$	$p_{21}(k)$	$p_{22}(k)$	$p_{23}(k)$	Value of Expression	
1	3,000	0	0	$\frac{1}{2}$	$\frac{1}{2}$	20,140	
2	4,000	0	1	0	0	18,636	← Minimum
3	6,000	1	0	0	0	19,454	

Since decision 1 minimizes the expression for state 1 and decision 2 minimizes the expression for state 2, our next trial policy (R_2) is as follows:

Policy R_2

State	Decision
0	1
1	1
2	2
3	3

Since this policy is identical to policy R_1, the optimality test indicates that this policy is optimal. Thus, the optimal policy under the average cost criterion also is optimal under the discounted cost criterion in this case. (This often occurs, but not always.)

Linear Programming Formulation

The linear programming formulation for the discounted cost case is similar to that for the average cost case given in Sec. 19.3. However, we no longer need the first constraint given in Sec. 19.3; but the other functional constraints do need to include the discount factor α. The other difference is that the model now contains constants β_j for $j = 0, 1, \ldots, M$. These constants must satisfy the conditions

$$\sum_{j=0}^{M} \beta_j = 1, \qquad \beta_j > 0 \qquad \text{for } j = 0, 1, \ldots, M,$$

but otherwise they can be chosen arbitrarily without affecting the optimal policy obtained from the model.

The resulting model is to choose the values of the *continuous* decision variables y_{ik} so as to

$$\text{Minimize} \qquad Z = \sum_{i=0}^{M} \sum_{k=1}^{K} C_{ik} y_{ik},$$

subject to the constraints

$$(1) \qquad \sum_{k=1}^{K} y_{jk} - \alpha \sum_{i=0}^{M} \sum_{k=1}^{K} y_{ik} p_{ij}(k) = \beta_j, \qquad \text{for } j = 0, 1, \ldots, M,$$

$$(2) \qquad y_{ik} \geq 0, \qquad \text{for } i = 0, 1, \ldots, M; \; k = 1, 2, \ldots, K.$$

Once the simplex method is used to obtain an optimal solution for this model, the corresponding optimal policy then is defined by

$$D_{ik} = P\{\text{decision} = k \,|\, \text{state} = i\} = \frac{y_{ik}}{\sum_{k=1}^{K} y_{ik}}.$$

The y_{ik} now can be interpreted as the *discounted* expected time of being in state i and making decision k, when the probability distribution of the *initial state* (when observations begin) is $P\{X_0 = j\} = \beta_j$ for $j = 0, 1, \ldots, M$. In other words, if

$$z_{ik}^n = P\{\text{at time } n, \text{ state} = i \text{ and decision} = k\},$$

then

$$y_{ik} = z_{ik}^0 + \alpha z_{ik}^1 + \alpha^2 z_{ik}^2 + \alpha^3 z_{ik}^3 + \cdots.$$

With the interpretation of the β_j as *initial state probabilities* (with each probability greater than zero), Z can be interpreted as the corresponding expected total discounted cost. Thus, the choice of β_j affects the optimal value of Z (but not the resulting optimal policy).

It again can be shown that the optimal policy obtained from solving the linear programming model is deterministic; that is, $D_{ik} = 0$ or 1. Furthermore, this technique is valid without the assumption (used for the average cost case) that the Markov chain associated with every transition matrix is irreducible.

The reason that this approach is attractive is that we already have a quick method of finding an optimal policy when the process has only n periods to go, namely, probabilistic dynamic programming as described in Sec. 10.4.

In particular, for $i = 0, 1, \ldots, M$, let

V_i^n = expected total discounted cost of following an optimal policy, given that process starts in state i and has only n periods to go.[1]

By the *principle of optimality* for dynamic programming (see Sec. 10.2), the V_i^n are obtained from the recursive relationship

$$V_i^n = \min_k \left\{ C_{ik} + \alpha \sum_{j=0}^{M} p_{ij}(k) V_j^{n-1} \right\}, \qquad \text{for } i = 0, 1, \ldots, M.$$

The minimizing value of k provides the optimal decision to make in the first period when the process starts in state i.

To get started, with $n = 1$, all the $V_i^0 = 0$ so that

$$V_i^1 = \min_k \{C_{ik}\}, \qquad \text{for } i = 0, 1, \ldots, M.$$

Although the method of successive approximations may not lead to an optimal policy for the infinite-period problem after only a few iterations, it has one distinct advantage over the policy improvement and linear programming techniques. It never requires solving a system of simultaneous equations, so each iteration can be performed simply and quickly.

Furthermore, if the Markov decision process actually does have just n periods to go, n iterations of this method definitely will lead to an optimal policy. (For an n-period problem, it is permissible to set $\alpha = 1$, that is, no discounting, in which case the objective is to minimize the expected total cost over n periods.)

Your IOR Tutorial includes an interactive procedure to help guide you to use this method efficiently.

Solving the Prototype Example by the Method of Successive Approximations

We again use $\alpha = 0.9$. Refer to the rightmost column of Table 19.1 at the end of Sec. 19.1 for the values of C_{ik}. Also note in the first two columns of this table that the only feasible decisions k for each state i are $k = 1$ for $i = 0$, $k = 1$ or 3 for $i = 1$, $k = 1$, 2, or 3 for $i = 2$, and $k = 3$ for $i = 3$.

For the first iteration ($n = 1$), the value obtained for each V_i^1 is shown below, along with the minimizing value of k (given in parentheses).

$$V_0^1 = \min_{k=1} \{C_{0k}\} = 0 \qquad (k = 1)$$
$$V_1^1 = \min_{k=1,3} \{C_{1k}\} = 1{,}000 \qquad (k = 1)$$
$$V_2^1 = \min_{k=1,2,3} \{C_{2k}\} = 3{,}000 \qquad (k = 1)$$
$$V_3^1 = \min_{k=3} \{C_{3k}\} = 6{,}000 \qquad (k = 3)$$

Thus, the first approximation calls for making decision 1 (do nothing) when the system is in state 0, 1, or 2. When the system is in state 3, decision 3 (replace the machine) is made.

[1]Since we want to allow n to grow indefinitely, we are letting n be the *number of periods to go*, instead of the *number of periods from the beginning* (as in Chap. 10).

The second iteration leads to

$$V_0^2 = 0 + 0.9\left[\frac{7}{8}(1{,}000) + \frac{1}{16}(3{,}000) + \frac{1}{16}(6{,}000)\right] = 1{,}294 \quad (k = 1)$$

$$V_1^2 = \min\left\{1{,}000 + 0.9\left[\frac{3}{4}(1{,}000) + \frac{1}{8}(3{,}000) + \frac{1}{8}(6{,}000)\right], 6{,}000 + 0.9[1(0)]\right\} = 2{,}688 \quad (k = 1)$$

$$V_2^2 = \min\left\{3{,}000 + 0.9\left[\frac{1}{2}(3{,}000) + \frac{1}{2}(6{,}000)\right], 4{,}000 + 0.9[1(1{,}000)], 6{,}000 + 0.9[1(0)]\right\} = 4{,}900 \quad (k = 2)$$

$$V_3^2 = 6{,}000 + 0.9[1(0)] = 6{,}000 \quad (k = 3).$$

where the *min* operator has been deleted from the first and fourth expressions because only one alternative for the decision is available. Thus, the second approximation calls for leaving the machine as is when it is in state 0 or 1, overhauling when it is in state 2, and replacing the machine when it is in state 3. Note that this policy is the optimal one for the infinite-period problem, as found earlier in this section by both the policy improvement algorithm and linear programming. However, the V_i^2 (the expected total discounted cost when starting in state i for the two-period problem) are not yet close to the V_i (the corresponding cost for the infinite-period problem).

The third iteration leads to

$$V_0^3 = 0 + 0.9\left[\frac{7}{8}(2{,}688) + \frac{1}{16}(4{,}900) + \frac{1}{16}(6{,}000)\right] = 2{,}730 \quad (k = 1)$$

$$V_1^3 = \min\left\{1{,}000 + 0.9\left[\frac{3}{4}(2{,}688) + \frac{1}{8}(4{,}900) + \frac{1}{8}(6{,}000)\right], 6{,}000 + 0.9[1(1{,}294)]\right\} = 4{,}041 \quad (k = 1)$$

$$V_2^3 = \min\left\{3{,}000 + 0.9\left[\frac{1}{2}(4{,}900) + \frac{1}{2}(6{,}000)\right], 4{,}000 + 0.9[1(2{,}688)], 6{,}000 + 0.9[1(1{,}294)]\right\} = 6{,}419 \quad (k = 2)$$

$$V_3^3 = 6{,}000 + 0.9[1(1{,}294)] = 7{,}165 \quad (k = 3).$$

Again the optimal policy for the infinite-period problem is obtained, and the costs are getting closer to those for that problem. This procedure can be continued, and V_0^n, V_1^n, V_2^n, and V_3^n will converge to 14,949, 16,262, 18,636, and 19,454, respectively.

Note that termination of the method of successive approximations after the second iteration would have resulted in an optimal policy for the infinite-period problem, although there is no way to know this fact without solving the problem by other methods.

As indicated earlier, the method of successive approximations definitely obtains an optimal policy for an n-period problem after n iterations. For this example, the first, second, and third iterations have identified the optimal immediate decision for each state if the remaining number of periods is one, two, and three, respectively.

19.6 CONCLUSIONS

Markov decision processes provide a powerful tool for optimizing the performance of stochastic processes that can be modeled as a discrete time Markov chain. Applications arise in a variety of areas, such as health care, highway and bridge maintenance, inventory

management, machine maintenance, cash-flow management, control of water reservoirs, forest management, control of queueing systems, and operation of communication networks. Selected References 10 and 11 provide interesting early surveys of applications. Selected Reference 9 gives an update on one that won a prestigious prize, and Selected Reference 4 describes another award-winning application. Selected References 3 and 7 include more recent information on applications.

The two primary measures of performance used are the (long-run) *expected average cost per unit time* and the *expected total discounted cost.* The latter measure requires determination of the appropriate value of a discount factor, but this measure is useful when it is important to take into account the time value of money.

The two most important methods for deriving optimal policies for Markov decision processes are *policy improvement algorithms* and *linear programming.* Under the discounted cost criterion, the *method of successive approximations* provides a quick way of approximating an optimal policy.

SELECTED REFERENCES

1. Bertsekas, D. P.: *Dynamic Programming: Deterministic and Stochastic Models,* Prentice-Hall, Englewood Cliffs, NJ, 1987.
2. Denardo, E.: *Dynamic Programming Theory and Applications,* Prentice-Hall, Englewood Cliffs, NJ, 1982.
3. Feinberg, E. A., and A. Shwartz: *Handbook of Markov Decision Processes: Methods and Applications,* Kluwer Academic Publishers, Boston, 2002.
4. Golabi, K., and R. Shepard: "Pontis: A System for Maintenance Optimization and Improvement of U.S. Bridge Networks," *Interfaces,* **27**(1): 71–88, January–February 1997.
5. Howard, R. A.: "Comments on the Origin and Application of Markov Decision Processes," *Operations Research,* **50:** 100–102, January–February 2002.
6. Puterman, M. L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming,* Wiley, New York, 1994.
7. Sennott, L. I.: *Stochastic Dynamic Programming and the Control of Queueing Systems,* Wiley, New York, 1999.
8. Smith J. E., and K. F. McCardle: "Structural Properties of Stochastic Dynamic Programs," *Operations Research,* **50:** 796–809, 2002.
9. Wang, K. C. P., and J. P. Zaniewski: "20/30 Hindsight: The New Pavement Optimization in the Arizona State Highway Network," *Interfaces,* **26**(3): 77–89, May–June 1996.
10. White, D. J.: "Further Real Applications of Markov Decision Processes," *Interfaces,* **18**(5): 55–61, September–October 1988.
11. White, D. J.: "Real Applications of Markov Decision Processes," *Interfaces,* **15**(6): 73–83, November–December 1985.
12. Whittle, P.: *Optimization over Time: Dynamic Programming and Stochastic Control,* Wiley, New York, vol. 1, 1982; vol. 2, 1983.

LEARNING AIDS FOR THIS CHAPTER ON CD-ROM

Worked Examples:

Examples for Chapter 19

A Demonstration Example in OR Tutor:

Policy Improvement Algorithm—Average Cost Case

Interactive Procedures in IOR Tutorial:

Enter Markov Decision Model
Interactive Policy Improvement Algorithm—Average Cost
Interactive Policy Improvement Algorithm—Discounted Cost
Interactive Method of Successive Approximations

Automatic Procedures in IOR Tutorial (Markov Chains Area):

Enter Transition Matrix
Steady-State Probabilities

"Ch. 19—Markov Decision Proc" Files for Solving the Linear Programming Formulations:

Excel Files
LINGO/LINDO File

Glossary for Chapter 19

See Appendix 1 for documentation of the software.

■ PROBLEMS

The symbols to the left of some of the problems (or their parts) have the following meaning:

D: The demonstration example listed on p. 925 may be helpful.
I: We suggest that you use the corresponding interactive procedure listed above (the printout records your work).
A: The automatic procedures listed above can be helpful.
C: Use the computer with any of the software options available to you (or as instructed by your instructor) to solve your linear programming formulation.

An asterisk on the problem number indicates that at least a partial answer is given in the back of the book.

19.2-1.* During any period, a potential customer arrives at a certain facility with probability $\frac{1}{2}$. If there are already two people at the facility (including the one being served), the potential customer leaves the facility immediately and never returns. However, if there is one person or less, he enters the facility and becomes an actual customer. The manager of the facility has two types of service configurations available. At the beginning of each period, a decision must be made on which configuration to use. If she uses her "slow" configuration at a cost of \$3 and any customers are present during the period, one customer will be served and leave the facility with probability $\frac{3}{5}$. If she uses her "fast" configuration at a cost of \$9 and any customers are present during the period, one customer will be served and leave the facility with probability $\frac{4}{5}$. The probability of more than one customer arriving or more than one customer being served in a period is zero. A profit of \$50 is earned when a customer is served.

(a) Formulate the problem of choosing the service configuration period by period as a Markov decision process. Identify the states and decisions. For each combination of state and decision, find the *expected net immediate cost* (subtracting any profit from serving a customer) incurred during that period.
(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average net cost per period in terms of the unknown steady-state probabilities ($\pi_0, \pi_1, \ldots, \pi_M$).
A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-2.* A student is concerned about her car and does not like dents. When she drives to school, she has a choice of parking it on the street in one space, parking it on the street and taking up two spaces, or parking in the lot. If she parks on the street in one space, her car gets dented with probability $\frac{1}{10}$. If she parks on the street and takes two spaces, the probability of a dent is $\frac{1}{50}$ and the probability of a \$15 ticket is $\frac{3}{10}$. Parking in a lot costs \$5, but the car will not get dented. If her car gets dented, she can have it repaired, in which case it is out of commission for 1 day and costs her \$50 in fees and cab fares. She can also drive her car dented, but she feels that the resulting loss of value and pride is equivalent to a cost of \$9 per school day. She wishes to determine the optimal policy for where to park and whether to repair the car when dented in order to minimize her (long-run) expected average cost per school day.
(a) Formulate this problem as a Markov decision process by identifying the states and decisions and then finding the C_{ik}.

(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average cost per period in terms of the unknown steady-state probabilities $(\pi_0, \pi_1, \ldots, \pi_M)$.

A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-3. Every Saturday night a man plays poker at his home with the same group of friends. If he provides refreshments for the group (at an expected cost of \$14) on any given Saturday night, the group will begin the following Saturday night in a good mood with probability $\frac{7}{8}$ and in a bad mood with probability $\frac{1}{8}$. However, if he fails to provide refreshments, the group will begin the following Saturday night in a good mood with probability $\frac{1}{8}$ and in a bad mood with probability $\frac{7}{8}$, regardless of their mood this Saturday. Furthermore, if the group begins the night in a bad mood and then he fails to provide refreshments, the group will gang up on him so that he incurs expected poker losses of \$75. Under other circumstances, he averages no gain or loss on his poker play. The man wishes to find the policy regarding when to provide refreshments that will minimize his (long-run) expected average cost per week.

(a) Formulate this problem as a Markov decision process by identifying the states and decisions and then finding the C_{ik}.

(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average cost per period in terms of the unknown steady-state probabilities $(\pi_0, \pi_1, \ldots, \pi_M)$.

A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-4.* When a tennis player serves, he gets two chances to serve in bounds. If he fails to do so twice, he loses the point. If he attempts to serve an ace, he serves in bounds with probability $\frac{3}{8}$. If he serves a lob, he serves in bounds with probability $\frac{7}{8}$. If he serves an ace in bounds, he wins the point with probability $\frac{2}{3}$. With an in-bounds lob, he wins the point with probability $\frac{1}{3}$. If the cost is $+1$ for each point lost and -1 for each point won, the problem is to determine the optimal serving strategy to minimize the (long-run) expected average cost per point. (*Hint:* Let state 0 denote point over, two serves to go on next point; and let state 1 denote one serve left.)

(a) Formulate this problem as a Markov decision process by identifying the states and decisions and then finding the C_{ik}.

(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average cost per point in terms of the unknown steady-state probabilities $(\pi_0, \pi_1, \ldots, \pi_M)$.

A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-5. Each year Ms. Fontanez has the chance to invest in two different no-load mutual funds: the Go-Go Fund or the Go-Slow Mutual Fund. At the end of each year, Ms. Fontanez liquidates her holdings, takes her profits, and then reinvests. The yearly profits of the mutual funds are dependent upon how the market reacts each year. Recently the market has been oscillating around the 12,000 mark from one year end to the next, according to the probabilities given in the following transition matrix:

	11,000	12,000	13,000
11,000	0.3	0.5	0.2
12,000	0.1	0.5	0.4
13,000	0.2	0.4	0.4

Each year that the market moves up (down) 1,000 points, the Go-Go Fund has profits (losses) of \$20,000, while the Go-Slow Fund has profits (losses) of \$10,000. If the market moves up (down) 2,000 points in a year, the Go-Go Fund has profits (losses) of \$50,000, while the Go-Slow Fund has profits (losses) of only \$20,000. If the market does not change, there is no profit or loss for either fund. Ms. Fontanez wishes to determine her optimal investment policy in order to minimize her (long-run) expected average cost (loss minus profit) per year.

(a) Formulate this problem as a Markov decision process by identifying the states and decisions and then finding the C_{ik}.

(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average cost per period in terms of the unknown steady-state probabilities $(\pi_0, \pi_1, \ldots, \pi_M)$.

A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-6. Buck and Bill Bogus are twin brothers who work at a gas station and have a counterfeiting business on the side. Each day a decision is made as to which brother will go to work at the gas station, and then the other will stay home and run the printing press in the basement. Each day that the machine works properly, it is estimated that 60 usable \$20 bills can be produced. However, the machine is somewhat unreliable and breaks down frequently. If the machine is not working at the beginning of the day, Buck can have it in working order by the beginning of the next day with probability 0.6. If Bill works on the machine, the probability decreases to 0.5. If Bill operates the machine when it is working, the probability is 0.6 that it will still be working at the beginning of the next day. If Buck operates the machine, it breaks down with probability 0.6. (Assume for simplicity that all breakdowns occur at the end of the day.) The brothers now wish to determine the optimal policy for when each should stay home in order to maximize their (long-run) expected average *profit* (amount of usable counterfeit money produced) per day.

(a) Formulate this problem as a Markov decision process by identifying the states and decisions and then finding the C_{ik}.

(b) Identify all the (stationary deterministic) policies. For each one, find the transition matrix and write an expression for the (long-run) expected average net profit per period in terms of the unknown steady-state probabilities $(\pi_0, \pi_1, \ldots, \pi_M)$.

A **(c)** Use your IOR Tutorial to find these steady-state probabilities for each policy. Then evaluate the expression obtained in part (*b*) to find the optimal policy by exhaustive enumeration.

19.2-7. Consider an infinite-period inventory problem involving a single product where, at the beginning of each period, a decision must be made about how many items to produce during that period. The setup cost is \$10, and the unit production cost is \$5. The holding cost for each item not sold during the period is \$4 (a *maximum* of 2 items can be stored). The demand during each period has a known probability distribution, namely, a probability of $\frac{1}{3}$ of 0, 1, and 2 items, respectively. If the demand exceeds the supply available during the period, then those sales are lost and a shortage cost (including lost revenue) is incurred, namely, \$8 and \$32 for a shortage of 1 and 2 items, respectively.

(a) Consider the policy where 2 items are produced if there are no items in inventory at the beginning of a period whereas no items are produced if there are any items in inventory. Determine the (long-run) expected average cost per period for this policy. In finding the transition matrix for the Markov chain for this policy, let the states represent the inventory levels at the beginning of the period.

(b) Identify all the *feasible* (stationary deterministic) inventory policies, i.e., the policies that never lead to exceeding the storage capacity.

19.3-1. Reconsider Prob. 19.2-1.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-2.* Reconsider Prob. 19.2-2.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-3. Reconsider Prob. 19.2-3.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-4.* Reconsider Prob. 19.2-4.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-5. Reconsider Prob. 19.2-5.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-6. Reconsider Prob. 19.2-6.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

19.3-7. Reconsider Prob. 19.2-7.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

D,I **19.4-1.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-1.

D,I **19.4-2.*** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-2.

D,I **19.4-3.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-3.

D,I **19.4-4.*** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-4.

D,I **19.4-5.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-5.

D,I **19.4-6.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-6.

D,I **19.4-7.** Use the policy improvement algorithm to find an optimal policy for Prob. 19.2-7.

D,I **19.4-8.** Consider the blood-inventory problem presented in Prob. 16.5-5. Suppose now that the number of pints of blood delivered (on a regular delivery) can be specified at the time of delivery (instead of using the old policy of receiving 1 pint at each delivery). Thus, the number of pints delivered can be 0, 1, 2, or 3 (more than 3 pints can never be used). The cost of regular delivery is \$50 per pint, while the cost of an emergency delivery is \$100 per pint. Starting with the proposed policy given in Prob. 16.5-5, perform two iterations of the policy improvement algorithm.

I **19.5-1.*** Joe wants to sell his car. He receives one offer each month and must decide immediately whether to accept the offer. Once rejected, the offer is lost. The possible offers are \$600, \$800, and \$1,000, made with probabilities $\frac{5}{8}$, $\frac{1}{4}$, and $\frac{1}{8}$, respectively (where successive offers are independent of each other). There is a maintenance cost of \$60 per month for the car. Joe is anxious to sell the car and so has chosen a discount factor of $\alpha = 0.95$.

Using the policy improvement algorithm, find a policy that minimizes the expected total discounted cost. (*Hint:* There are two actions: Accept or reject the offer. Let the state for month t be the offer in that month. Also include a state ∞, where the process goes to state ∞ whenever an offer is accepted and it remains there at a monthly cost of 0.)

19.5-2.* Reconsider Prob. 19.5-1.

(a) Formulate a linear programming model for finding an optimal policy.

C **(b)** Use the simplex method to solve this model. Use the resulting optimal solution to identify an optimal policy.

I **19.5-3.*** For Prob. 19.5-1, use three iterations of the method of successive approximations to approximate an optimal policy.