Deterministic Dynamic Programming

Dynamic programming is a technique that can be used to solve many optimization problems. In most applications, dynamic programming obtains solutions by working backward from the end of a problem toward the beginning, thus breaking up a large, unwieldy problem into a series of smaller, more tractable problems.

We introduce the idea of working backward by solving two well-known puzzles and then show how dynamic programming can be used to solve network, inventory, and resource-allocation problems. We close the chapter by showing how to use spreadsheets to solve dynamic programming problems.

18.1 Two Puzzles[†]

In this section, we show how working backward can make a seemingly difficult problem almost trivial to solve.

EXAMPLE 1

Suppose there are 30 matches on a table. I begin by picking up 1, 2, or 3 matches. Then my opponent must pick up 1, 2, or 3 matches. We continue in this fashion until the last match is picked up. The player who picks up the last match is the loser. How can I (the first player) be sure of winning the game?

Solution If I can ensure that it will be my opponent's turn when 1 match remains, I will certainly win. Working backward one step, if I can ensure that it will be my opponent's turn when 5 matches remain, I will win. The reason for this is that no matter what he does when 5 matches remain, I can make sure that when he has his next turn, only 1 match will remain. For example, suppose it is my opponent's turn when 5 matches remain. If my opponent picks up 2 matches, I will pick up 2 matches, leaving him with 1 match and sure defeat. Similarly, if I can force my opponent to play when 5, 9, 13, 17, 21, 25, or 29 matches remain, I am sure of victory. Thus, I cannot lose if I pick up $30 - 29 = 1$ match on my first turn. Then I simply make sure that my opponent will always be left with 29, 25, 21, 17, 13, 9, or 5 matches on his turn. Notice that we have solved this puzzle by working backward from the end of the problem toward the beginning. Try solving this problem without working backward!

EXAMPLE 2

I have a 9-oz cup and a 4-oz cup. My mother has ordered me to bring home exactly 6 oz of milk. How can I accomplish this goal?

[†]This section covers topics that may be omitted with no loss of continuity.

TABLE 1

Moves in the Cup-and-Milk Problem

No. of Ounces in 9-oz Cup	No. of Ounces in 4-oz Cup
6	0
6	4
9	1
0	1
1	0
1	4
5	0
5	4
9	0
0	0

Solution By starting near the end of the problem, I cleverly realize that the problem can easily be solved if I can somehow get 1 oz of milk into the 4-oz cup. Then I can fill the 9-oz cup and empty 3 oz from the 9-oz cup into the partially filled 4-oz cup. At this point, I will be left with 6 oz of milk. After I have this flash of insight, the solution to the problem may easily be described as in Table 1 (the initial situation is written last, and the final situation is written first).

PROBLEMS

Group A

1 Suppose there are 40 matches on a table. I begin by picking up 1, 2, 3, or 4 matches. Then my opponent must pick up 1, 2, 3, or 4 matches. We continue until the last match is picked up. The player who picks up the last match is the loser. Can I be sure of victory? If so, how?

2 Three players have played three rounds of a gambling game. Each round has one loser and two winners. The losing player must pay each winner the amount of money that the winning player had at the beginning of the round. At the end of the three rounds each player has $10. You are told that each player has won one round. By working backward, determine the original stakes of the three players. [*Note:* If the answer turns out to be (for example) 5, 15, 10, don't worry about which player had which stake; we can't really tell which player ends up with how much, but we can determine the numerical values of the original stakes.]

Group B

3 We have 21 coins and are told that one is heavier than any of the other coins. How many weighings on a balance will it take to find the heaviest coin? (*Hint:* If the heaviest coin is in a group of three coins, we can find it in one weighing. Then work backward to two weighings, and so on.)

4 Given a 7-oz cup and a 3-oz cup, explain how we can return from a well with 5 oz of water.

18.2 A Network Problem

Many applications of dynamic programming reduce to finding the shortest (or longest) path that joins two points in a given network. The following example illustrates how dynamic programming (working backward) can be used to find the shortest path in a network.

EXAMPLE 3 Shortest Path

Joe Cougar lives in New York City, but he plans to drive to Los Angeles to seek fame and fortune. Joe's funds are limited, so he has decided to spend each night on his trip at a friend's house. Joe has friends in Columbus, Nashville, Louisville, Kansas City, Omaha, Dallas, San Antonio, and Denver. Joe knows that after one day's drive he can reach Columbus, Nashville, or Louisville. After two days of driving, he can reach Kansas City, Omaha, or Dallas. After three days of driving, he can reach San Antonio or Denver. Finally, after four days of driving, he can reach Los Angeles. To minimize the number of miles traveled, where should Joe spend each night of the trip? The actual road mileages between cities are given in Figure 1.

Solution Joe needs to know the shortest path between New York and Los Angeles in Figure 1. We will find it by working backward. We have classified all the cities that Joe can be in at the beginning of the *n*th day of his trip as stage *n* cities. For example, because Joe can only be in San Antonio or Denver at the beginning of the fourth day (day 1 begins when Joe leaves New York), we classify San Antonio and Denver as stage 4 cities. The reason for classifying cities according to stages will become apparent later.

The idea of working backward implies that we should begin by solving an easy problem that will eventually help us to solve a complex problem. Hence, we begin by finding the shortest path to Los Angeles from each city in which there is only one day of driving left (stage 4 cities). Then we use this information to find the shortest path to Los Angeles from each city for which only two days of driving remain (stage 3 cities). With this information in hand, we are able to find the shortest path to Los Angeles from each city that is three days distant (stage 2 cities). Finally, we find the shortest path to Los Angeles from each city (there is only one: New York) that is four days away.

To simplify the exposition, we use the numbers 1, 2, . . . , 10 given in Figure 1 to label the 10 cities. We also define c_{ij} to be the road mileage between city i and city j. For example, $c_{35} = 580$ is the road mileage between Nashville and Kansas City. We let $f_t(i)$ be the length of the shortest path from city i to Los Angeles, given that city i is a stage t city.[†]

Stage 4 Computations

We first determine the shortest path to Los Angeles from each stage 4 city. Since there is only one path from each stage 4 city to Los Angeles, we immediately see that $f_4(8) = 1{,}030$, the shortest path from Denver to Los Angeles simply being the *only* path from Denver to Los Angeles. Similarly, $f_4(9) = 1{,}390$, the shortest (and only) path from San Antonio to Los Angeles.

Stage 3 Computations

We now work backward one stage (to stage 3 cities) and find the shortest path to Los Angeles from each stage 3 city. For example, to determine $f_3(5)$, we note that the shortest path from city 5 to Los Angeles must be one of the following:

Path 1 Go from city 5 to city 8 and then take the shortest path from city 8 to city 10.

Path 2 Go from city 5 to city 9 and then take the shortest path from city 9 to city 10.

The length of path 1 may be written as $c_{58} + f_4(8)$, and the length of path 2 may be written as $c_{59} + f_4(9)$. Hence, the shortest distance from city 5 to city 10 may be written as

[†]In this example, keeping track of the stages is unnecessary; to be consistent with later examples, however, we do keep track.

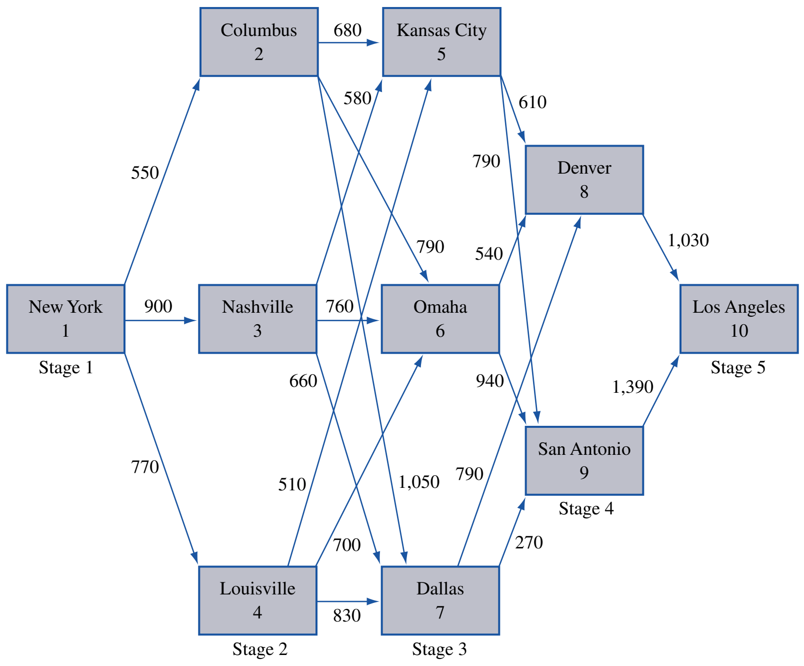


FIGURE 1
Joe's Trip Across the United States

$$f_3(5) = \min \begin{cases} c_{58} + f_4(8) = 610 + 1{,}030 = 1{,}640^* \\ c_{59} + f_4(9) = 790 + 1{,}390 = 2{,}180 \end{cases}$$

[the * indicates the choice of arc that attains the $f_3(5)$]. Thus, we have shown that the shortest path from city 5 to city 10 is the path 5–8–10. Note that to obtain this result, we made use of our knowledge of $f_4(8)$ and $f_4(9)$.

Similarly, to find $f_3(6)$, we note that the shortest path to Los Angeles from city 6 must begin by going to city 8 or to city 9. This leads us to the following equation:

$$f_3(6) = \min \begin{cases} c_{68} + f_4(8) = 540 + 1{,}030 = 1{,}570^* \\ c_{69} + f_4(9) = 940 + 1{,}390 = 2{,}330 \end{cases}$$

Thus, $f_3(6) = 1{,}570$, and the shortest path from city 6 to city 10 is the path 6–8–10.

To find $f_3(7)$, we note that

$$f_3(7) = \min \begin{cases} c_{78} + f_4(8) = 790 + 1{,}030 = 1{,}820 \\ c_{79} + f_4(9) = 270 + 1{,}390 = 1{,}660^* \end{cases}$$

Therefore, $f_3(7) = 1{,}660$, and the shortest path from city 7 to city 10 is the path 7–9–10.

Stage 2 Computations

Given our knowledge of $f_3(5)$, $f_3(6)$, and $f_3(7)$, it is now easy to work backward one more stage and compute $f_2(2)$, $f_2(3)$, and $f_2(4)$ and thus the shortest paths to Los Angeles from city 2, city 3, and city 4. To illustrate how this is done, we find the shortest path (and its length) from city 2 to city 10. The shortest path from city 2 to city 10 must begin by going from city 2 to city 5, city 6, or city 7. Once this shortest path gets to city 5, city 6, or city 7, then it must follow a shortest path from that city to Los Angeles. This reasoning shows that the shortest path from city 2 to city 10 must be one of the following:

Path 1 Go from city 2 to city 5. Then follow a shortest path from city 5 to city 10. A path of this type has a total length of $c_{25} + f_3(5)$.

Path 2 Go from city 2 to city 6. Then follow a shortest path from city 6 to city 10. A path of this type has a total length of $c_{26} + f_3(6)$.

Path 3 Go from city 2 to city 7. Then follow a shortest path from city 7 to city 10. This path has a total length of $c_{27} + f_3(7)$. We may now conclude that

$$f_2(2) = \min \begin{cases} c_{25} + f_3(5) = 680 + 1{,}640 = 2{,}320^* \\ c_{26} + f_3(6) = 790 + 1{,}570 = 2{,}360 \\ c_{27} + f_3(7) = 1{,}050 + 1{,}660 = 2{,}710 \end{cases}$$

Thus, $f_2(2) = 2{,}320$, and the shortest path from city 2 to city 10 is to go from city 2 to city 5 and then follow the shortest path from city 5 to city 10 (5–8–10).

Similarly,

$$f_2(3) = \min \begin{cases} c_{35} + f_3(5) = 580 + 1{,}640 = 2{,}220^* \\ c_{36} + f_3(6) = 760 + 1{,}570 = 2{,}330 \\ c_{37} + f_3(7) = 660 + 1{,}660 = 2{,}320 \end{cases}$$

Thus, $f_2(3) = 2{,}220$, and the shortest path from city 3 to city 10 consists of arc 3–5 and the shortest path from city 5 to city 10 (5–8–10).

In similar fashion,

$$f_2(4) = \min \begin{cases} c_{45} + f_3(5) = 510 + 1{,}640 = 2{,}150^* \\ c_{46} + f_3(6) = 700 + 1{,}570 = 2{,}270 \\ c_{47} + f_3(7) = 830 + 1{,}660 = 2{,}490 \end{cases}$$

Thus, $f_2(4) = 2{,}150$, and the shortest path from city 4 to city 10 consists of arc 4–5 and the shortest path from city 5 to city 10 (5–8–10).

Stage 1 Computations

We can now use our knowledge of $f_2(2)$, $f_2(3)$, and $f_2(4)$ to work backward one more stage to find $f_1(1)$ and the shortest path from city 1 to city 10. Note that the shortest path from city 1 to city 10 must begin by going to city 2, city 3, or city 4. This means that the shortest path from city 1 to city 10 must be one of the following:

Path 1 Go from city 1 to city 2 and then follow a shortest path from city 2 to city 10. The length of such a path is $c_{12} + f_2(2)$.

Path 2 Go from city 1 to city 3 and then follow a shortest path from city 3 to city 10. The length of such a path is $c_{13} + f_2(3)$.

Path 3 Go from city 1 to city 4 and then follow a shortest path from city 4 to city 10. The length of such a path is $c_{14} + f_2(4)$. It now follows that

$$f_1(1) = \min \begin{cases} c_{12} + f_2(2) = 550 + 2{,}320 = 2{,}870^* \\ c_{13} + f_2(3) = 900 + 2{,}220 = 3{,}120 \\ c_{14} + f_2(4) = 770 + 2{,}150 = 2{,}920 \end{cases}$$

Determination of the Optimal Path

Thus, $f_1(1) = 2{,}870$, and the shortest path from city 1 to city 10 goes from city 1 to city 2 and then follows the shortest path from city 2 to city 10. Checking back to the $f_2(2)$ calculations, we see that the shortest path from city 2 to city 10 is 2–5–8–10. Translating the numerical labels into real cities, we see that the shortest path from New York to Los Angeles passes through New York, Columbus, Kansas City, Denver, and Los Angeles. This path has a length of $f_1(1) = 2{,}870$ miles.

Computational Efficiency of Dynamic Programming

For Example 3, it would have been an easy matter to determine the shortest path from New York to Los Angeles by enumerating all the possible paths [after all, there are only $3(3)(2) = 18$ paths]. Thus, in this problem, the use of dynamic programming did not really serve much purpose. For larger networks, however, dynamic programming is much more efficient for determining a shortest path than the explicit enumeration of all paths. To see this, consider the network in Figure 2. In this network, it is possible to travel from any node in stage k to any node in stage $k + 1$. Let the distance between node i and node j be c_{ij}. Suppose we want to determine the shortest path from node 1 to node 27. One way to solve this problem is explicit enumeration of all paths. There are 5^5 possible paths from node 1 to node 27. It takes five additions to determine the length of each path. Thus, explicitly enumerating the length of all paths requires $5^5(5) = 5^6 = 15{,}625$ additions.

Suppose we use dynamic programming to determine the shortest path from node 1 to node 27. Let $f_t(i)$ be the length of the shortest path from node i to node 27, given that node i is in stage t. To determine the shortest path from node 1 to node 27, we begin by finding $f_6(22)$, $f_6(23)$, $f_6(24)$, $f_6(25)$, and $f_6(26)$. This does not require any additions. Then we find $f_5(17)$, $f_5(18)$, $f_5(19)$, $f_5(20)$, $f_5(21)$. For example, to find $f_5(21)$ we use the following equation:

$$f_5(21) = \min_j \{c_{21,j} + f_6(j)\} \qquad (j = 22, 23, 24, 25, 26)$$

Determining $f_5(21)$ in this manner requires five additions. Thus, the calculation of all the $f_5(\cdot)$'s requires $5(5) = 25$ additions. Similarly, the calculation of all the $f_4(\cdot)$'s requires 25 additions, and the calculation of all the $f_3(\cdot)$'s requires 25 additions. The determination of all the $f_2(\cdot)$'s also requires 25 additions, and the determination of $f_1(1)$ requires 5 additions. Thus, in total, dynamic programming requires $4(25) + 5 = 105$ additions to find

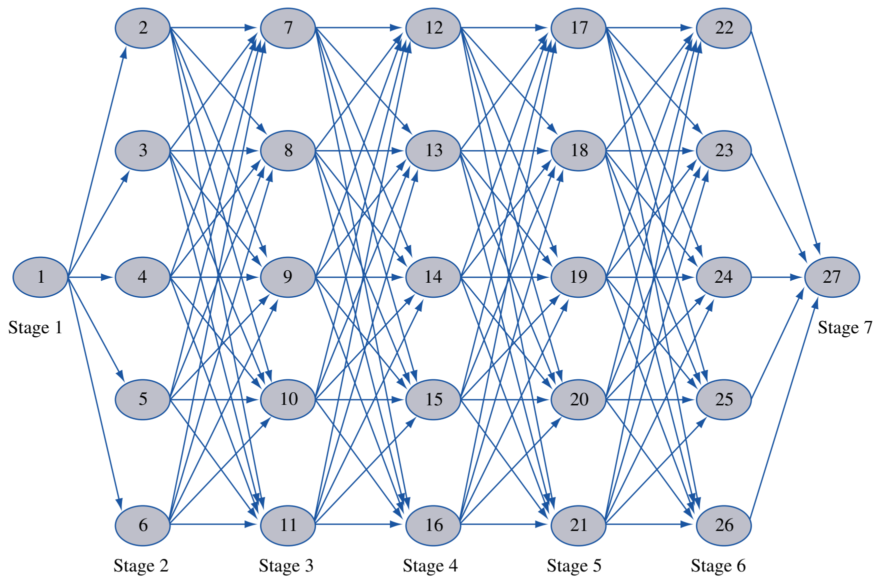


FIGURE 2
Illustration of Computational Efficiency of Dynamic Programming

the shortest path from node 1 to node 27. Because explicit enumeration requires 15,625 additions, we see that dynamic programming requires only 0.007 times as many additions as explicit enumeration. For larger networks, the computational savings effected by dynamic programming are even more dramatic.

Besides additions, determination of the shortest path in a network requires comparisons between the lengths of paths. If explicit enumeration is used, then $5^5 - 1 = 3{,}124$ comparisons must be made (that is, compare the length of the first two paths, then compare the length of the third path with the shortest of the first two paths, and so on). If dynamic programming is used, then for $t = 2, 3, 4, 5$, determination of each $f_t(i)$ requires $5 - 1 = 4$ comparisons. Then to compute $f_1(1)$, $5 - 1 = 4$ comparisons are required. Thus, to find the shortest path from node 1 to node 27, dynamic programming requires a total of $20(5 - 1) + 4 = 84$ comparisons. Again, dynamic programming comes out far superior to explicit enumeration.

Characteristics of Dynamic Programming Applications

We close this section with a discussion of the characteristics of Example 3 that are common to most applications of dynamic programming.

Characteristic 1

The problem can be divided into stages with a decision required at each stage. In Example 3, stage t consisted of those cities where Joe could be at the beginning of day t of his trip. As we will see, in many dynamic programming problems, the stage is the amount of time that has elapsed since the beginning of the problem. We note that in some situations, decisions are not required at every stage (see Section 18.5).

Characteristic 2

Each stage has a number of states associated with it. By a **state,** we mean the information that is needed at any stage to make an optimal decision. In Example 3, the state at stage t is simply the city where Joe is at the beginning of day t. For example, in stage 3, the possible states are Kansas City, Omaha, and Dallas. Note that to make the correct decision at any stage, Joe doesn't need to know how he got to his current location. For example, if Joe is in Kansas City, then his remaining decisions don't depend on how he goes to Kansas City; his future decisions just depend on the fact that he is now in Kansas City.

Characteristic 3

The decision chosen at any stage describes how the state at the current stage is transformed into the state at the next stage. In Example 3, Joe's decision at any stage is simply the next city to visit. This determines the state at the next stage in an obvious fashion. In many problems, however, a decision does not determine the next stage's state with certainty; instead, the current decision only determines the probability distribution of the state at the next stage.

Characteristic 4

Given the current state, the optimal decision for each of the remaining stages must not depend on previously reached states or previously chosen decisions. This idea is known as the **principle of optimality.** In the context of Example 3, the principle of optimality reduces to the following: Suppose the shortest path (call it R) from city 1 to city 10 is known to pass through city i. Then the portion of R that goes from city i to city 10 must be a shortest path from city i to city 10. If this were not the case, then we could create a path from city 1 to city 10 that was shorter than R by appending a shortest path from city i to city 10 to the portion of R leading from city 1 to city i. This would create a path from city 1 to city 10 that is shorter than R, thereby contradicting the fact that R is a shortest path from city 1 to city 10. For example, if the shortest path from city 1 to city 10 is known to pass through city 2, then the shortest path from city 1 to city 10 must include a shortest path from city 2 to city 10 (2–5–8–10). This follows because any path from city 1 to city 10 that passes through city 2 and does not contain a shortest path from city 2 to city 10 will have a length of c_{12} + [something bigger than $f_2(2)$]. Of course, such a path cannot be a shortest path from city 1 to city 10.

Characteristic 5

If the states for the problem have been classified into one of T stages, there must be a recursion that relates the cost or reward earned during stages t, t + 1, . . . , *T to the cost or reward earned from stages t* + 1, *t* + 2, . . . , *T.* In essence, the recursion formalizes the working-backward procedure. In Example 3, our recursion could have been written as

$$f_t(i) = \min_{j} \{c_{ij} + f_{t+1}(j)\}$$

where j must be a stage $t + 1$ city and $f_5(10) = 0$.

We can now describe how to make optimal decisions. Let's assume that the initial state during stage 1 is i_1. To use the recursion, we begin by finding the optimal decision for each state associated with the last stage. Then we use the recursion described in characteristic 5 to determine $f_{T-1}(\cdot)$ (along with the optimal decision) for every stage $T - 1$ state. Then we use the recursion to determine $f_{T-2}(\cdot)$ (along with the optimal decision) for every stage $T - 2$ state. We continue in this fashion until we have computed $f_1(i_1)$ and the optimal decision when we are in stage 1 and state i_1. Then our optimal decision in stage 1 is chosen from the set of decisions attaining $f_1(i_1)$. Choosing this decision at stage 1 will lead us to some stage 2 state (call it state i_2) at stage 2. Then at stage 2, we choose any decision attaining $f_2(i_2)$. We continue in this fashion until a decision has been chosen for each stage.

In the rest of this chapter, we discuss many applications of dynamic programming. The presentation will seem easier if the reader attempts to determine how each problem fits into the network context introduced in Example 3. In the next section, we begin by studying how dynamic programming can be used to solve inventory problems.

PROBLEMS

Group A

1 Find the shortest path from node 1 to node 10 in the network shown in Figure 3. Also, find the shortest path from node 3 to node 10.

2 A sales representative lives in Bloomington and must be in Indianapolis next Thursday. On each of the days Monday, Tuesday, and Wednesday, he can sell his wares in Indianapolis, Bloomington, or Chicago. From past experience, he believes that he can earn \$12 from spending a day in Indianapolis, \$16 from spending a day in Bloomington, and \$17 from spending a day in Chicago. Where should he spend the first three days

FIGURE 3

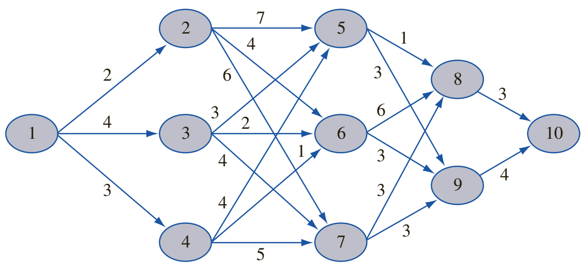


TABLE 2

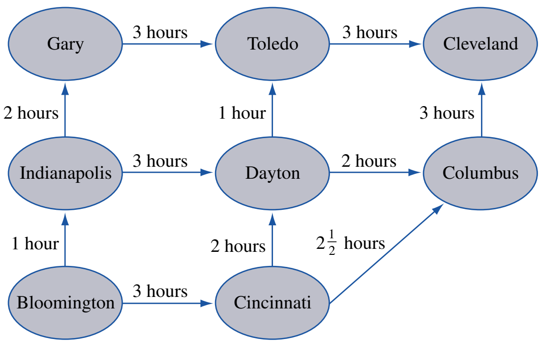
From	To		
	Indianapolis	Bloomington	Chicago
Indianapolis	—	5	2
Bloomington	5	—	7
Chicago	2	7	—

and nights of the week to maximize his sales income less travel costs? Travel costs are shown in Table 2.

Group B

3 I must drive from Bloomington to Cleveland. Several paths are available (see Figure 4). The number on each arc is the length of time it takes to drive between the two cities. For example, it takes 3 hours to drive from Bloomington to Cincinnati. By working backward, determine the shortest path (in terms of time) from Bloomington to Cleveland. [*Hint:* Work backward and don't worry about stages—only about states.]

FIGURE 4



18.3 An Inventory Problem

In this section, we illustrate how dynamic programming can be used to solve an inventory problem with the following characteristics:

1 Time is broken up into periods, the present period being period 1, the next period 2, and the final period T. At the beginning of period 1, the demand during each period is known.

2 At the beginning of each period, the firm must determine how many units should be produced. Production capacity during each period is limited.

3 Each period's demand must be met on time from inventory or current production. During any period in which production takes place, a fixed cost of production as well as a variable per-unit cost is incurred.

4 The firm has limited storage capacity. This is reflected by a limit on end-of-period inventory. A per-unit holding cost is incurred on each period's ending inventory.

5 The firm's goal is to minimize the total cost of meeting on time the demands for periods 1, 2, . . . , T.

In this model, the firm's inventory position is reviewed at the end of each period (say, at the end of each month), and then the production decision is made. Such a model is called a **periodic review model.** This model is in contrast to the continuous review models in which the firm knows its inventory position at all times and may place an order or begin production at any time.

If we exclude the setup cost for producing any units, the inventory problem just described is similar to the Sailco inventory problem that we solved by linear programming in Section 3.10. Here, we illustrate how dynamic programming can be used to determine a production schedule that minimizes the total cost incurred in an inventory problem that meets the preceding description.

EXAMPLE 4 Inventory

A company knows that the demand for its product during each of the next four months will be as follows: month 1, 1 unit; month 2, 3 units; month 3, 2 units; month 4, 4 units. At the beginning of each month, the company must determine how many units should be produced during the current month. During a month in which any units are produced, a setup cost of \$3 is incurred. In addition, there is a variable cost of \$1 for every unit produced. At the end of each month, a holding cost of 50¢ per unit on hand is incurred. Capacity limitations allow a maximum of 5 units to be produced during each month. The size of the company's warehouse restricts the ending inventory for each month to 4 units at most. The company wants to determine a production schedule that will meet all demands on time and will minimize the sum of production and holding costs during the four months. Assume that 0 units are on hand at the beginning of the first month.

Solution Recall from Section 3.10 that we can ensure that all demands are met on time by restricting each month's ending inventory to be nonnegative. To use dynamic programming to solve this problem, we need to identify the appropriate state, stage, and decision. The stage should be defined so that when one stage remains, the problem will be trivial to solve. If we are at the beginning of month 4, then the firm would meet demand at minimum cost by simply producing just enough units to ensure that (month 4 production) + (month 3 ending inventory) = (month 4 demand). Thus, when one month remains, the firm's problem is easy to solve. Hence, we let time represent the stage. In most dynamic programming problems, the stage has something to do with time.

At each stage (or month), the company must decide how many units to produce. To make this decision, the company need only know the inventory level at the beginning of the current month (or the end of the previous month). Therefore, we let the state at any stage be the beginning inventory level.

Before writing a recursive relation that can be used to "build up" the optimal production schedule, we must define $f_t(i)$ to be the minimum cost of meeting demands for months $t, t + 1, \dots, 4$ if i units are on hand at the beginning of month t. We define $c(x)$ to be the cost of producing x units during a period. Then $c(0) = 0$, and for $x > 0$, $c(x) = 3 + x$. Because of the limited storage capacity and the fact that all demand must be met on time, the possible states during each period are 0, 1, 2, 3, and 4. Thus, we begin by determining $f_4(0)$, $f_4(1)$, $f_4(2)$, $f_4(3)$, and $f_4(4)$. Then we use this information to determine $f_3(0)$, $f_3(1)$, $f_3(2)$, $f_3(3)$, and $f_3(4)$. Then we determine $f_2(0)$, $f_2(1)$, $f_2(2)$, $f_2(3)$, and $f_2(4)$. Finally, we determine $f_1(0)$. Then we determine an optimal production level for each month. We define $x_t(i)$ to be a production level during month t that minimizes the total cost during months $t, t + 1, \dots, 4$ if i units are on hand at the beginning of month t. We now begin to work backward.

Month 4 Computations

During month 4, the firm will produce just enough units to ensure that the month 4 demand of 4 units is met. This yields

$$f_4(0) = \text{cost of producing } 4 - 0 \text{ units} = c(4) = 3 + 4 = \$7 \text{ and } x_4(0) = 4 - 0 = 4$$
$$f_4(1) = \text{cost of producing } 4 - 1 \text{ units} = c(3) = 3 + 3 = \$6 \text{ and } x_4(1) = 4 - 1 = 3$$
$$f_4(2) = \text{cost of producing } 4 - 2 \text{ units} = c(2) = 3 + 2 = \$5 \text{ and } x_4(2) = 4 - 2 = 2$$
$$f_4(3) = \text{cost of producing } 4 - 3 \text{ units} = c(1) = 3 + 1 = \$4 \text{ and } x_4(3) = 4 - 3 = 1$$
$$f_4(4) = \text{cost of producing } 4 - 4 \text{ units} = c(0) = \$0 \quad \text{and} \quad x_4(4) = 4 - 4 = 0$$

Month 3 Computations

How can we now determine f_3(i) for $i = 0, 1, 2, 3, 4$? The cost $f_3(i)$ is the minimum cost incurred during months 3 and 4 if the inventory at the beginning of month 3 is i. For each possible production level x during month 3, the total cost during months 3 and 4 is

$$(\tfrac{1}{2})(i + x - 2) + c(x) + f_4(i + x - 2) \tag{1}$$

This follows because if x units are produced during month 3, the ending inventory for month 3 will be $i + x - 2$. Then the month 3 holding cost will be $(\frac{1}{2})(i + x - 2)$, and the month 3 production cost will be $c(x)$. Then we enter month 4 with $i + x - 2$ units on hand. Since we proceed optimally from this point onward (remember the principle of optimality), the cost for month 4 will be $f_4(i + x - 2)$. We want to choose the month 3 production level to minimize (1), so we write

$$f_3(i) = \min_x \{(\tfrac{1}{2})(i + x - 2) + c(x) + f_4(i + x - 2)\} \tag{2}$$

In (2), x must be a member of $\{0, 1, 2, 3, 4, 5\}$, and x must satisfy $4 \geq i + x - 2 \geq 0$. This reflects the fact that the current month's demand must be met $(i + x - 2 \geq 0)$, and ending inventory cannot exceed the capacity of $4(i + x - 2 \leq 4)$. Recall that $x_3(i)$ is any value of x attaining $f_3(i)$. The computations for $f_3(0)$, $f_3(1)$, $f_3(2)$, $f_3(3)$, and $f_3(4)$ are given in Table 3.

Month 2 Computations

We can now determine $f_2(i)$, the minimum cost incurred during months 2, 3, and 4 given that at the beginning of month 2, the on-hand inventory is i units. Suppose that month 2 production $= x$. Because month 2 demand is 3 units, a holding cost of $(\frac{1}{2})(i + x - 3)$ is

TABLE 3
Computations for $f_3(i)$

i	x	$(\frac{1}{2})(i + x - 2) + c(x)$	$f_4(i + x - 2)$	Total Cost Months 3, 4	$f_3(i)$ $x_3(i)$
0	2	$0 + 5 = 5$	7	$5 + 7 = 12$*	$f_3(0) = 12$
0	3	$\frac{1}{2} + 6 = \frac{13}{2}$	6	$\frac{13}{2} + 6 = \frac{25}{2}$	$x_3(0) = 2$
0	4	$1 + 7 = 8$	5	$8 + 5 = 13$	
0	5	$\frac{3}{2} + 8 = \frac{19}{2}$	4	$\frac{19}{2} + 4 = \frac{27}{2}$	
1	1	$0 + 4 = 4$	7	$4 + 7 = 11$	$f_3(1) = 10$
1	2	$\frac{1}{2} + 5 = \frac{11}{2}$	6	$\frac{11}{2} + 6 = \frac{23}{2}$	$x_3(1) = 5$
1	3	$1 + 6 = 7$	5	$7 + 5 = 12$	
1	4	$\frac{3}{2} + 7 = \frac{17}{2}$	4	$\frac{17}{2} + 4 = \frac{25}{2}$	
1	5	$2 + 8 = 10$	0	$10 + 0 = 10$*	
2	0	$0 + 0 = 0$	7	$0 + 7 = 7$*	$f_3(2) = 7$
2	1	$\frac{1}{2} + 4 = \frac{9}{2}$	6	$\frac{9}{2} + 6 = \frac{21}{2}$	$x_3(2) = 0$
2	2	$1 + 5 = 6$	5	$6 + 5 = 11$	
2	3	$\frac{3}{2} + 6 = \frac{15}{2}$	4	$\frac{15}{2} + 4 = \frac{23}{2}$	
2	4	$2 + 7 = 9$	0	$9 + 0 = 9$	
3	0	$\frac{1}{2} + 0 = \frac{1}{2}$	6	$\frac{1}{2} + 6 = \frac{13}{2}$*	$f_3(3) = \frac{13}{2}$
3	1	$1 + 4 = 5$	5	$5 + 5 = 10$	$x_3(3) = 0$
3	2	$\frac{3}{2} + 5 = \frac{13}{2}$	4	$\frac{13}{2} + 4 = \frac{21}{2}$	
3	3	$2 + 6 = 8$	0	$8 + 0 = 8$	
4	0	$1 + 0 = 1$	5	$1 + 5 = 6$*	$f_3(4) = 6$
4	1	$\frac{3}{2} + 4 = \frac{11}{2}$	4	$\frac{11}{2} + 4 = \frac{19}{2}$	$x_3(4) = 0$
4	2	$2 + 5 = 7$	0	$7 + 0 = 7$	

incurred at the end of month 2. Thus, the total cost incurred during month 2 is $(\frac{1}{2})(i + x - 3) + c(x)$. During months 3 and 4, we follow an optimal policy. Since month 3 begins with an inventory of $i + x - 3$, the cost incurred during months 3 and 4 is $f_3(i + x - 3)$. In analogy to (2), we now write

$$f_2(i) = \min_x \{(\tfrac{1}{2})(i + x - 3) + c(x) + f_3(i + x - 3)\} \tag{3}$$

where x must be a member of $\{0, 1, 2, 3, 4, 5\}$ and x must also satisfy $0 \leq i + x - 3 \leq 4$. The computations for $f_2(0)$, $f_2(1)$, $f_2(2)$, $f_2(3)$, and $f_2(4)$ are given in Table 4.

Month 1 Computations

The reader should now be able to show that the $f_1(i)$'s can be determined via the following recursive relation:

$$f_1(i) = \min_x \{(\tfrac{1}{2})(i + x - 1) + c(x) + f_2(i + x - 1)\} \tag{4}$$

where x must be a member of $\{0, 1, 2, 3, 4, 5\}$ and x must satisfy $0 \leq i + x - 1 \leq 4$. Since the inventory at the beginning of month 1 is 0 units, we actually need only determine $f_1(0)$ and $x_1(0)$. To give the reader more practice, however, the computations for $f_1(1)$, $f_1(2)$, $f_1(3)$, and $f_1(4)$ are given in Table 5.

Determination of the Optimal Production Schedule

We can now determine a production schedule that minimizes the total cost of meeting the demand for all four months on time. Since our initial inventory is 0 units, the minimum cost for the four months will be $f_1(0) = \$20$. To attain $f_1(0)$, we must produce $x_1(0) = 1$

TABLE 4
Computations for $f_2(i)$

i	x	$(\frac{1}{2})(i + x - 3) + c(x)$	$f_3(i + x - 3)$	Total Cost Months 2–4	$f_2(i)$ $x_2(i)$
0	3	$0 + 6 = 6$	12	$6 + 12 = 18$	$f_2(0) = 16$
0	4	$\frac{1}{2} + 7 = \frac{15}{2}$	10	$\frac{15}{2} + 10 = \frac{35}{2}$	$x_2(0) = 5$
0	5	$1 + 8 = 9$	7	$9 + 7 = 16^*$	
1	2	$0 + 5 = 5$	12	$5 + 12 = 17$	$f_2(1) = 15$
1	3	$\frac{1}{2} + 6 = \frac{13}{2}$	10	$\frac{13}{2} + 10 = \frac{33}{2}$	$x_2(1) = 4$
1	4	$1 + 7 = 8$	7	$8 + 7 = 15^*$	
1	5	$\frac{3}{2} + 8 = \frac{19}{2}$	$\frac{13}{2}$	$\frac{19}{2} + \frac{13}{2} = 16$	
2	1	$0 + 4 = 4$	12	$4 + 12 = 16$	$f_2(2) = 14$
2	2	$\frac{1}{2} + 5 = \frac{11}{2}$	10	$\frac{11}{2} + 10 = \frac{31}{2}^*$	$x_2(2) = 3$
2	3	$1 + 6 = 7$	7	$7 + 7 = 14^*$	
2	4	$\frac{3}{2} + 7 = \frac{17}{2}$	$\frac{13}{2}$	$\frac{17}{2} + \frac{13}{2} = 15$	
2	5	$2 + 8 = 10$	6	$10 + 6 = 16$	
3	0	$0 + 0 = 0$	12	$0 + 12 = 12^*$	$f_2(3) = 12$
3	1	$\frac{1}{2} + 4 = \frac{9}{2}$	10	$\frac{9}{2} + 10 = \frac{29}{2}$	$x_2(3) = 0$
3	2	$1 + 5 = 6$	7	$6 + 7 = 13$	
3	3	$\frac{3}{2} + 6 = \frac{15}{2}$	$\frac{13}{2}$	$\frac{15}{2} + \frac{13}{2} = 14$	
3	4	$2 + 7 = 9$	6	$9 + 6 = 15$	
4	0	$\frac{1}{2} + 0 = \frac{1}{2}$	10	$\frac{1}{2} + 10 = \frac{21}{2}^*$	$f_2(4) = \frac{21}{2}$
4	1	$1 + 4 = 5$	7	$5 + 7 = 12$	$x_2(4) = 0$
4	2	$\frac{3}{2} + 5 = \frac{13}{2}$	$\frac{13}{2}$	$\frac{13}{2} + \frac{13}{2} = 13$	
4	3	$2 + 6 = 8$	6	$8 + 6 = 14$	

TABLE 5

Computations for $f_1(i)$

i	x	$(\frac{1}{2})(i+x-1)+c(x)$	$f_2(i+x-1)$	Total Cost	$f_1(i)$ $x_1(i)$
0	1	$0+4=4$	16	$4+16=20^*$	$f_1(0)=20$
0	2	$\frac{1}{2}+5=\frac{11}{2}$	15	$\frac{11}{2}+15=\frac{41}{2}$	$x_1(0)=1$
0	3	$1+6=7$	14	$7+14=21$	
0	4	$\frac{3}{2}+7=\frac{17}{2}$	12	$\frac{17}{2}+12=\frac{41}{2}$	
0	5	$2+8=10$	$\frac{21}{2}$	$10+\frac{21}{2}=\frac{41}{2}$	
1	0	$0+0=0$	16	$0+16=16^*$	$f_1(1)=16$
1	1	$\frac{1}{2}+4=\frac{9}{2}$	15	$\frac{9}{2}+15=\frac{39}{2}$	$x_1(1)=0$
1	2	$1+5=6$	14	20	
1	3	$\frac{3}{2}+6=\frac{15}{2}$	12	$\frac{15}{2}+12=\frac{39}{2}$	
1	4	$2+7=9$	$\frac{21}{2}$	$9+\frac{21}{2}=\frac{39}{2}$	
2	0	$\frac{1}{2}+0=\frac{1}{2}$	15	$\frac{1}{2}+15=\frac{31}{2}^*$	$f_1(2)=\frac{31}{2}$
2	1	$1+4=5$	14	$5+14=19$	$x_1(2)=0$
2	2	$\frac{3}{2}+5=\frac{13}{2}$	12	$\frac{13}{2}+12=\frac{37}{2}$	
2	3	$2+6=8$	$\frac{21}{2}$	$8+\frac{21}{2}=\frac{37}{2}$	
3	0	$1+0=1$	14	$1+14=15^*$	$f_1(3)=15$
3	1	$\frac{3}{2}+4=\frac{11}{2}$	12	$\frac{11}{2}+12=\frac{35}{2}$	$x_1(3)=0$
3	2	$2+5=7$	$\frac{21}{2}$	$7+\frac{21}{2}=\frac{35}{2}$	
4	0	$\frac{3}{2}+0=\frac{3}{2}$	12	$\frac{3}{2}+12=\frac{27}{2}^*$	$f_1(4)=\frac{27}{2}$
4	1	$2+4=6$	$\frac{21}{2}$	$6+\frac{21}{2}=\frac{33}{2}$	$x_1(4)=0$

unit during month 1. Then the inventory at the beginning of month 2 will be $0 + 1 - 1 = 0$. Thus, in month 2, we should produce $x_2(0) = 5$ units. Then at the beginning of month 3, our beginning inventory will be $0 + 5 - 3 = 2$. Hence, during month 3, we need to produce $x_3(2) = 0$ units. Then month 4 will begin with $2 - 2 + 0 = 0$ units on hand. Thus, $x_4(0) = 4$ units should be produced during month 4. In summary, the optimal production schedule incurs a total cost of \$20 and produces 1 unit during month 1, 5 units during month 2, 0 units during month 3, and 4 units during month 4.

Note that finding the solution to Example 4 is equivalent to finding the shortest route joining the node (1, 0) to the node (5, 0) in Figure 5. Each node in Figure 5 corresponds to a state, and each column of nodes corresponds to all the possible states associated with a given stage. For example, if we are at node (2, 3), then we are at the beginning of month 2, and the inventory at the beginning of month 2 is 3 units. Each arc in the network represents the way in which a decision (how much to produce during the current month) transforms the current state into next month's state. For example, the arc joining nodes (1, 0) and (2, 2) (call it arc 1) corresponds to producing 3 units during month 1. To see this, note that if 3 units are produced during month 1, then we begin month 2 with $0 + 3 - 1 = 2$ units. The length of each arc is simply the sum of production and inventory costs during the current period, given the current state and the decision associated with the chosen arc. For example, the cost associated with arc 1 would be $6 + (\frac{1}{2})2 = 7$. Note that some nodes in adjacent stages are not joined by an arc. For example, node (2, 4) is not joined to node (3, 0). The reason for this is that if we begin month 2 with 4 units, then at the beginning of month 3, we will have at least $4 - 3 = 1$ unit on hand. Also note that we have drawn arcs joining all month 4 states to the node (5, 0), since having a positive inventory at the end of month 4 would clearly be suboptimal.

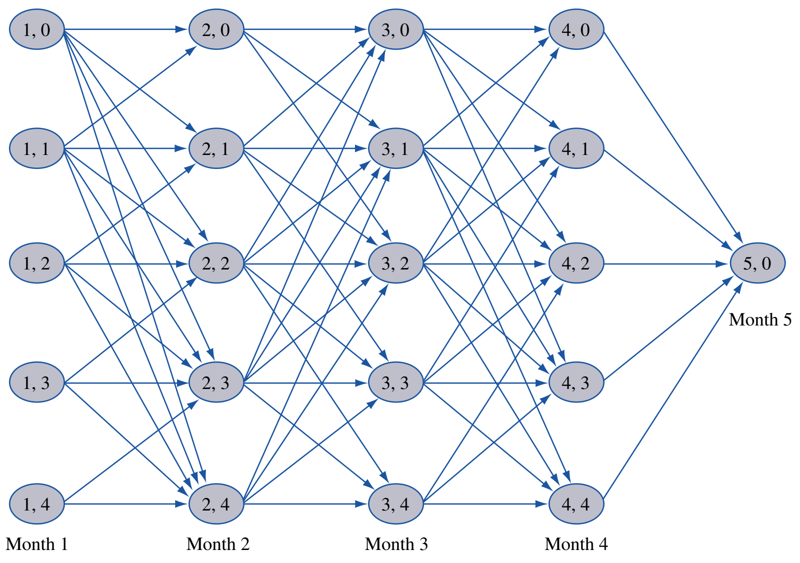


FIGURE 5
Network Representation of Inventory Example

Returning to Example 4, the minimum-cost production schedule corresponds to the shortest path joining (1, 0) and (5, 0). As we have already seen, this would be the path corresponding to production levels of 1, 5, 0, and 4. In Figure 5, this would correspond to the path beginning at (1, 0), then going to $(2, 0 + 1 - 1) = (2, 0)$, then to $(3, 0 + 5 - 3) = (3, 2)$, then to $(4, 2 + 0 - 2) = (4, 0)$, and finally to $(5, 0 + 4 - 4) = (5, 0)$. Thus, our optimal production schedule corresponds to the path (1, 0)–(2, 0)–(3, 2)–(4, 0)–(5, 0) in Figure 5.

PROBLEMS

Group A

1 In Example 4, determine the optimal production schedule if the initial inventory is 3 units.

2 An electronics firm has a contract to deliver the following number of radios during the next three months; month 1, 200 radios; month 2, 300 radios; month 3, 300 radios. For each radio produced during months 1 and 2, a \$10 variable cost is incurred; for each radio produced during month 3, a \$12 variable cost is incurred. The inventory cost is \$1.50 for each radio in stock at the end of a month. The cost of setting up for production during a month is \$250. Radios made during a month may be used to meet demand for that month or any future month. Assume that production during each month must be a multiple of 100. Given that the initial inventory level is 0 units, use dynamic programming to determine an optimal production schedule.

3 In Figure 5, determine the production level and cost associated with each of the following arcs:

a (2, 3)–(3, 1)
b (4, 2)–(5, 0)

18.4 Resource-Allocation Problems

Resource-allocation problems, in which limited resources must be allocated among several activities, are often solved by dynamic programming. Recall that we have solved such problems by linear programming (for instance, the Giapetto problem). To use linear programming to do resource allocation, three assumptions must be made:

Assumption 1 The amount of a resource assigned to an activity may be any nonnegative number.

Assumption 2 The benefit obtained from each activity is proportional to the amount of the resource assigned to the activity.

Assumption 3 The benefit obtained from more than one activity is the sum of the benefits obtained from the individual activities.

Even if assumptions 1 and 2 do not hold, dynamic programming can be used to solve resource-allocation problems efficiently when assumption 3 is valid and when the amount of the resource allocated to each activity is a member of a finite set.

EXAMPLE 5 Resource Allocation

Finco has \$6,000 to invest, and three investments are available. If d_j dollars (in thousands) are invested in investment j, then a net present value (in thousands) of $r_j(d_j)$ is obtained, where the $r_j(d_j)$'s are as follows:

$$\begin{aligned} r_1(d_1) &= 7d_1 + 2 && (d_1 > 0)\\ r_2(d_2) &= 3d_2 + 7 && (d_2 > 0)\\ r_3(d_3) &= 4d_3 + 5 && (d_3 > 0)\\ r_1(0) &= r_2(0) = r_3(0) = 0 \end{aligned}$$

The amount placed in each investment must be an exact multiple of \$1,000. To maximize the net present value obtained from the investments, how should Finco allocate the \$6,000?

Solution The return on each investment is not proportional to the amount invested in it [for example, $16 = r_1(2) \neq 2r_1(1) = 18$]. Thus, linear programming cannot be used to find an optimal solution to this problem.†

Mathematically, Finco's problem may be expressed as

$$\begin{aligned} &\max\{r_1(d_1) + r_2(d_2) + r_3(d_3)\}\\ &\text{s.t.} \qquad d_1 + d_2 + d_3 = 6\\ &d_j \text{ nonnegative integer} \qquad (j = 1, 2, 3) \end{aligned}$$

Of course, if the $r_j(d_j)$'s were linear, then we would have a knapsack problem like those we studied in Section 9.5.

To formulate Finco's problem as a dynamic programming problem, we begin by identifying the stage. As in the inventory and shortest-route examples, the stage should be chosen so that when one stage remains the problem is easy to solve. Then, given that the problem has been solved for the case where one stage remains, it should be easy to solve the problem where two stages remain, and so forth. Clearly, it would be easy to solve when only one investment was available, so we define stage t to represent a case where funds must be allocated to investments $t, t + 1, \ldots, 3$.

For a given stage, what must we know to determine the optimal investment amount? Simply how much money is available for investments $t, t + 1, \ldots, 3$. Thus, we define the state at any stage to be the amount of money (in thousands) available for investments t, $t + 1, \ldots, 3$. We can never have more than \$6,000 available, so the possible states at any stage are 0, 1, 2, 3, 4, 5, and 6. We define $f_t(d_t)$ to be the maximum net present value (NPV) that can be obtained by investing d_t thousand dollars in investments $t, t + 1, \ldots,$ 3. Also define $x_t(d_t)$ to be the amount that should be invested in investment t to attain $f_t(d_t)$. We start to work backward by computing $f_3(0), f_3(1), \ldots, f_3(6)$ and then determine $f_2(0)$, $f_2(1), \ldots, f_2(6)$. Since \$6,000 is available for investment in investments 1, 2, and 3, we

†The fixed-charge approach described in Section 9.2 could be used to solve this problem.

terminate our computations by computing $f_1(6)$. Then we retrace our steps and determine the amount that should be allocated to each investment (just as we retraced our steps to determine the optimal production level for each month in Example 4).

Stage 3 Computations

We first determine $f_3(0), f_3(1), \ldots, f_3(6)$. We see that $f_3(d_3)$ is attained by investing all available money (d_3) in investment 3. Thus,

$$\begin{aligned} f_3(0) &= 0 & x_3(0) &= 0 \\ f_3(1) &= 9 & x_3(1) &= 1 \\ f_3(2) &= 13 & x_3(2) &= 2 \\ f_3(3) &= 17 & x_3(3) &= 3 \\ f_3(4) &= 21 & x_3(4) &= 4 \\ f_3(5) &= 25 & x_3(5) &= 5 \\ f_3(6) &= 29 & x_3(6) &= 6 \end{aligned}$$

TABLE 6
Computations for $f_2(0), f_2(1), \ldots, f_2(6)$

d_2	x_2	$r_2(x_2)$	$f_3(d_2 - x_2)$	NPV from Investments 2, 3	$f_2(d_2)$ $x_2(d_2)$
0	0	0	0	0*	$f_2(0) = 0$
					$x_2(0) = 0$
1	0	0	9	9	$f_2(1) = 10$
1	1	10	0	10*	$x_2(1) = 1$
2	0	0	13	13	$f_2(2) = 19$
2	1	10	9	19*	$x_2(2) = 1$
2	2	13	0	13	
3	0	0	17	17	$f_2(3) = 23$
3	1	10	13	23*	$x_2(3) = 1$
3	2	13	9	22	
3	3	16	0	16	
4	0	0	21	21	$f_2(4) = 27$
4	1	10	17	27*	$x_2(4) = 1$
4	2	13	13	26	
4	3	16	9	25	
4	4	19	0	19	
5	0	0	25	25	$f_2(5) = 31$
5	1	10	21	31*	$x_2(5) = 1$
5	2	13	17	30	
5	3	16	13	29	
5	4	19	9	28	
5	5	22	0	22	
6	0	0	29	29	$f_2(6) = 35$
6	1	10	25	35*	$x_2(6) = 1$
6	2	13	21	34	
6	3	16	17	33	
6	4	19	13	32	
6	5	22	9	31	
6	6	25	0	25	

TABLE 7
Computations for $f_1(6)$

d_1	x_1	$r_1(x_1)$	$f_2(6 - x_1)$	NPV from Investments 1–3	$f_1(6)$ $x_1(6)$
6	0	0	35	35	$f_1(6) = 49$
6	1	9	31	40	$x_1(6) = 4$
6	2	16	27	43	
6	3	23	23	46	
6	4	30	19	49*	
6	5	37	10	47	
6	6	44	0	44	

Stage 2 Computations

To determine $f_2(0), f_2(1), \dots, f_2(6)$, we look at all possible amounts that can be placed in investment 2. To find $f_2(d_2)$, let x_2 be the amount invested in investment 2. Then an NPV of $r_2(x_2)$ will be obtained from investment 2, and an NPV of $f_3(d_2 - x_2)$ will be obtained from investment 3 (remember the principle of optimality). Since x_2 should be chosen to maximize the net present value earned from investments 2 and 3, we write

$$f_2(d_2) = \max_{x_2}\{r_2(x_2) + f_3(d_2 - x_2)\} \tag{5}$$

where x_2 must be a member of $\{0, 1, \dots, d_2\}$. The computations for $f_2(0), f_2(1), \dots, f_2(6)$ and $x_2(0), x_2(1), \dots, x_2(6)$ are given in Table 6.

Stage 1 Computations

Following (5), we write

$$f_1(6) = \max_{x_1}\{r_1(x_1) + f_2(6 - x_1)\}$$

where x_1 must be a member of $\{0, 1, 2, 3, 4, 5, 6\}$. The computations for $f_1(6)$ are given in Table 7.

Determination of Optimal Resource Allocation

Since $x_1(6) = 4$, Finco invests \$4,000 in investment 1. This leaves $6{,}000 - 4{,}000 =$ \$2,000 for investments 2 and 3. Hence, Finco should invest $x_2(2) =$ \$1,000 in investment 2. Then \$1,000 is left for investment 3, so Finco chooses to invest $x_3(1) =$ \$1,000 in investment 3. Therefore, Finco can attain a maximum net present value of $f_1(6) =$ \$49,000 by investing \$4,000 in investment 1, \$1,000 in investment 2, and \$1,000 in investment 3.

Network Representation of Resource Example

As with the inventory example of Section 18.3, Finco's problem has a network representation, equivalent to finding the *longest route* from (1, 6) to (4, 0) in Figure 6. In the figure, the node (t, d) represents the situation in which d thousand dollars is available for investments $t, t + 1, \dots, 3$. The arc joining the nodes (t, d) and $(t + 1, d - x)$ has a length $r_t(x)$ corresponding to the net present value obtained by investing x thousand dollars in investment t. For example, the arc joining nodes (2, 4) and (3, 1) has a length $r_2(3) =$ \$16,000, corresponding to the \$16,000 net present value that can be obtained by invest-

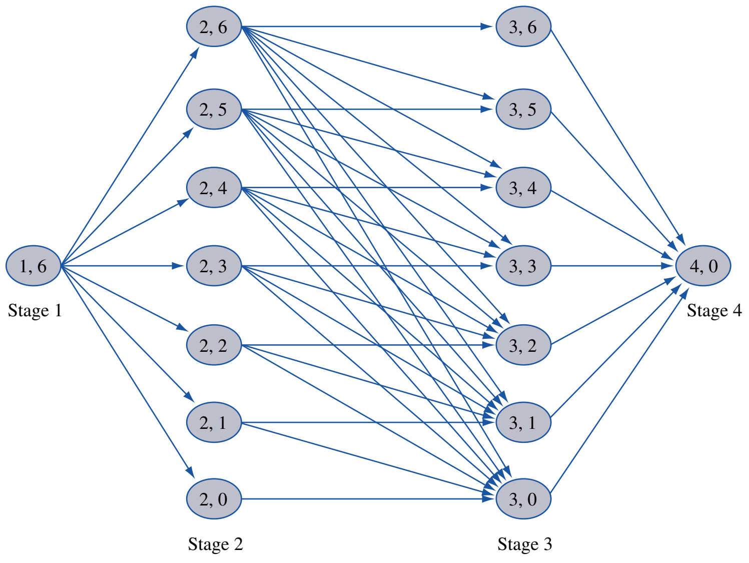


FIGURE 6
Network Representation of Finco

ing \$3,000 in investment 2. Note that not all pairs of nodes in adjacent stages are joined by arcs. For example, there is no arc joining the nodes (2, 4) and (3, 5); after all, if you have only \$4,000 available for investments 2 and 3, how can you have \$5,000 available for investment 3? From our computations, we see that the longest path from (1, 6) to (4, 0) is (1, 6)–(2, 2)–(3, 1)–(4, 0).

Generalized Resource Allocation Problem

We now consider a generalized version of Example 5. Suppose we have w units of a resource available and T activities to which the resource can be allocated. If activity t is implemented at a level x_t (we assume x_t must be a nonnegative integer), then $g_t(x_t)$ units of the resource are used by activity t, and a benefit $r_t(x_t)$ is obtained. The problem of determining the allocation of resources that maximizes total benefit subject to the limited resource availability may be written as

$$\begin{aligned} \max \sum_{t=1}^{t=T} r_t(x_t) \\ \text{s.t.} \quad \sum_{t=1}^{t=T} g_t(x_t) \le w \end{aligned} \tag{6}$$

where x_t must be a member of $\{0, 1, 2, \ldots\}$. Some possible interpretations of $r_t(x_t)$, $g_t(x_t)$, and w are given in Table 8.

To solve (6) by dynamic programming, define $f_t(d)$ to be the maximum benefit that can be obtained from activities $t, t + 1, \ldots, T$ if d units of the resource may be allocated to activities $t, t + 1, \ldots, T$. We may generalize the recursions of Example 5 to this situation by writing

$$\begin{aligned} f_{T+1}(d) &= 0 \quad \text{for all } d \\ f_t(d) &= \max_{x_t} \{r_t(x_t) + f_{t+1}[d - g_t(x_t)]\} \end{aligned} \tag{7}$$

where x_t must be a nonnegative integer satisfying $g_t(x_t) \le d$. Let $x_t(d)$ be any value of x_t that attains $f_t(d)$. To use (7) to determine an optimal allocation of resources to activities $1, 2, \ldots, T$, we begin by determining all $f_T(\cdot)$ and $x_T(\cdot)$. Then we use (7) to determine all $f_{T-1}(\cdot)$ and $x_{T-1}(\cdot)$, continuing to work backward in this fashion until all $f_2(\cdot)$ and $x_2(\cdot)$

TABLE 8

Examples of a Generalized Resource Allocation Problem

Interpretation of $r_t(x_t)$	Interpretation of $g_t(x_t)$	Interpretation of w
Benefit from placing x_t type t items in a knapsack	Weight of x_t type t items	Maximum weight that knapsack can hold
Grade obtained in course t if we study course t for x_t hours per week	Number of hours per week x_t spent studying course t	Total number of study hours available each week
Sales of a product in region t if x_t sales reps are assigned to region t	Cost of assigning x_t sales reps to region t	Total sales force budget
Number of fire alarms per week responded to within one minute if precinct t is assigned x_t engines	Cost per week of maintaining x_t fire engines in precinct t	Total weekly budget for maintaining fire engines

have been determined. To wind things up, we now calculate $f_1(w)$ and $x_1(w)$. Then we implement activity 1 at a level $x_1(w)$. At this point, we have $w - g_1[x_1(w)]$ units of the resource available for activities 2, 3, . . . , T. Then activity 2 should be implemented at a level of $x_2\{w - g_1[x_1(w)]\}$. We continue in this fashion until we have determined the level at which all activities should be implemented.

Solution of Knapsack Problems by Dynamic Programming

We illustrate the use of (7) by solving a simple knapsack problem (see Section 9.5). Then we develop an alternative recursion that can be used to solve knapsack problems.

EXAMPLE 6 Knapsack

Suppose a 10-lb knapsack is to be filled with the items listed in Table 9. To maximize total benefit, how should the knapsack be filled?

Solution We have $r_1(x_1) = 11x_1$, $r_2(x_2) = 7x_2$, $r_3(x_3) = 12x_3$, $g_1(x_1) = 4x_1$, $g_2(x_2) = 3x_2$, and $g_3(x_3) = 5x_3$. Define $f_t(d)$ to be the maximum benefit that can be earned from a d-pound knapsack that is filled with items of Type t, $t + 1$, . . . , 3.

Stage 3 Computations

Now (7) yields

$$f_3(d) = \max_{x_3}\{12x_3\}$$

TABLE 9

Weights and Benefits for Knapsack

Item	Weight (lb)	Benefit
1	4	11
2	3	7
3	5	12

where $5x_3 \le d$ and x_3 is a nonnegative integer. This yields

$$\begin{aligned} f_3(10) &= 24 \\ f_3(5) = f_3(6) = f_3(7) &= f_3(8) = f_3(9) = 12 \\ f_3(0) = f_3(1) = f_3(2) &= f_3(3) = f_3(4) = 0 \\ x_3(10) &= 2 \\ x_3(9) = x_3(8) = x_3(7) &= x_3(6) = x_3(5) = 1 \\ x_3(0) = x_3(1) = x_3(2) &= x_3(3) = x_3(4) = 0 \end{aligned}$$

Stage 2 Computations

Now (7) yields

$$f_2(d) = \max_{x_2} \{7x_2 + f_3(d - 3x_2)\}$$

where x_2 must be a nonnegative integer satisfying $3x_2 \le d$. We now obtain

$$f_2(10) = \max \begin{cases} 7(0) + f_3(10) = 24^* & x_2 = 0 \\ 7(1) + f_3(7) = 19 & x_2 = 1 \\ 7(2) + f_3(4) = 14 & x_2 = 2 \\ 7(3) + f_3(1) = 21 & x_2 = 3 \end{cases}$$

Thus, $f_2(10) = 24$ and $x_2(10) = 0$.

$$f_2(9) = \max \begin{cases} 7(0) + f_3(9) = 12 & x_2 = 0 \\ 7(1) + f_3(6) = 19 & x_2 = 1 \\ 7(2) + f_3(3) = 14 & x_2 = 2 \\ 7(3) + f_3(0) = 21^* & x_2 = 3 \end{cases}$$

Thus, $f_2(9) = 21$ and $x_2(9) = 3$.

$$f_2(8) = \max \begin{cases} 7(0) + f_3(8) = 12 & x_2 = 0 \\ 7(1) + f_3(5) = 19^* & x_2 = 1 \\ 7(2) + f_3(2) = 14 & x_2 = 2 \end{cases}$$

Thus, $f_2(8) = 19$ and $x_2(8) = 1$.

$$f_2(7) = \max \begin{cases} 7(0) + f_3(7) = 12 & x_2 = 0 \\ 7(1) + f_3(4) = 7 & x_2 = 1 \\ 7(2) + f_3(1) = 14^* & x_2 = 2 \end{cases}$$

Thus, $f_2(7) = 14$ and $x_2(7) = 2$.

$$f_2(6) = \max \begin{cases} 7(0) + f_3(6) = 12 & x_2 = 0 \\ 7(1) + f_3(3) = 7 & x_2 = 1 \\ 7(2) + f_3(0) = 14^* & x_2 = 2 \end{cases}$$

Thus, $f_2(6) = 14$ and $x_2(6) = 2$.

$$f_2(5) = \max \begin{cases} 7(0) + f_3(5) = 12^* & x_2 = 0 \\ 7(1) + f_3(2) = 7 & x_2 = 1 \end{cases}$$

Thus, $f_2(5) = 12$ and $x_2(5) = 0$.

$$f_2(4) = \max \begin{cases} 7(0) + f_3(4) = 0 & x_2 = 0 \\ 7(1) + f_3(1) = 7^* & x_2 = 1 \end{cases}$$

Thus, $f_2(4) = 7$ and $x_2(4) = 1$.

$$f_2(3) = \max \begin{cases} 7(0) + f_3(3) = 0 & x_2 = 0 \\ 7(1) + f_3(0) = 7^* & x_2 = 1 \end{cases}$$

Thus, $f_2(3) = 7$ and $x_2(3) = 1$.

$$f_2(2) = 7(0) + f_3(2) = 0 \qquad x_2 = 0$$

Thus, $f_2(2) = 0$ and $x_2(2) = 0$.

$$f_2(1) = 7(0) + f_3(1) = 0 \qquad x_2 = 0$$

Thus, $f_2(1) = 0$ and $x_2(1) = 0$.

$$f_2(0) = 7(0) + f_3(0) = 0 \qquad x_2 = 0$$

Thus, $f_2(0) = 0$ and $x_2(0) = 0$.

Stage 1 Computations

Finally, we determine $f_1(10)$ from

$$f_1(10) = \max \begin{cases} 11(0) + f_2(10) = 24 & x_1 = 0 \\ 11(1) + f_2(6) = 25^* & x_1 = 1 \\ 11(2) + f_2(2) = 22 & x_1 = 2 \end{cases}$$

Determination of the Optimal Solution to Knapsack Problem

We have $f_1(10) = 25$ and $x_1(10) = 1$. Hence, we should include one Type 1 item in the knapsack. Then we have $10 - 4 = 6$ lb left for Type 2 and Type 3 items, so we should include $x_2(6) = 2$ Type 2 items. Finally, we have $6 - 2(3) = 0$ lb left for Type 3 items, and we include $x_3(0) = 0$ Type 3 items. In summary, the maximum benefit that can be gained from a 10-lb knapsack is $f_3(10) = 25$. To obtain a benefit of 25, one Type 1 and two Type 2 items should be included.

Network Representation of Knapsack Problem

Finding the optimal solution to Example 6 is equivalent to finding the longest path in Figure 7 from node (10, 1) to some stage 4 node. In Figure 7, for $t \leq 3$, the node (d, t) represents a situation in which d pounds of space may be allocated to items of Type $t, t + 1, \ldots, 3$. The node $(d, 4)$ represents d pounds of unused space. Each arc from a stage t node to a stage $t + 1$ node represents a decision of how many Type t items are placed in the knapsack. For example, the arc from (10, 1) to (6, 2) represents placing one Type 1 item in the knapsack. This leaves $10 - 4 = 6$ lb for items of Types 2 and 3. This arc has a length of 11, representing the benefit obtained by placing one Type 1 item in the knapsack. Our solution to Example 6 shows that the longest path in Figure 7 from node (10, 1) to a stage 4 node is (10, 1)–(6, 2)–(0, 3)–(0, 4). We note that the optimal solution to a knapsack problem does not always use all the available weight. For example, the reader should verify that if a Type 1 item earned 16 units of benefit, the optimal solution would be to include two type 1 items, corresponding to the path (10, 1)–(2, 2)–(2, 3)–(2, 4). This solution leaves 2 lb of space unused.

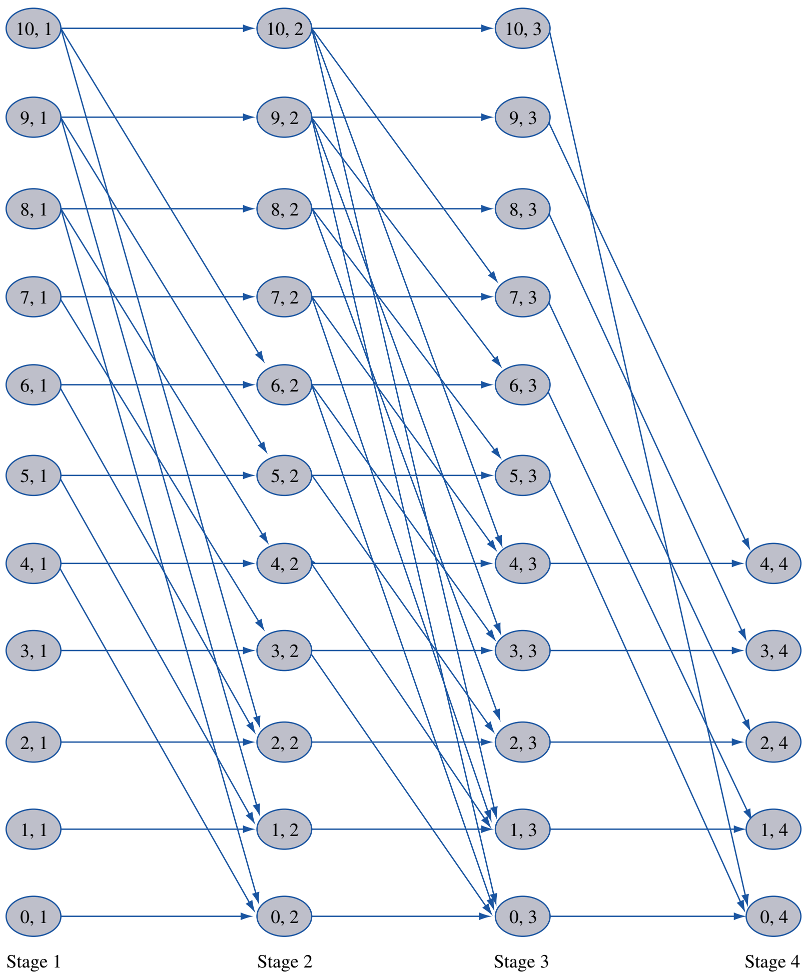


FIGURE 7
Network of Representation of Knapsack

An Alternative Recursion for Knapsack Problems

Other approaches can be used to solve knapsack problems by dynamic programming. The approach we now discuss builds up the optimal knapsack by first determining how to fill a small knapsack optimally and then, using this information, how to fill a larger knapsack optimally. We define $g(w)$ to be the maximum benefit that can be gained from a w-lb knapsack. In what follows, b_j is the benefit earned from a single Type j item, and w_j is the weight of a single Type j item. Clearly, $g(0) = 0$, and for $w > 0$,

$$g(w) = \max_{j} \{b_j + g(w - w_j)\} \tag{8}$$

where j must be a member of $\{1, 2, 3\}$, and j must satisfy $w_j \leq w$. The reasoning behind (8) is as follows: To fill a w-lb knapsack optimally, we must begin by putting some type of item into the knapsack. If we begin by putting a Type j item into a w-lb knapsack, the best we can do is earn b_j + [best we can do from a $(w - w_j)$-lb knapsack]. After noting that a Type j item can be placed into a w-lb knapsack only if $w_j \leq w$, we obtain (8). We define $x(w)$ to be any type of item that attains the maximum in (8) and $x(w) = 0$ to mean that no item can fit into a w-lb knapsack.

To illustrate the use of (8), we re-solve Example 6. Because no item can fit in a 0-, 1-, or 2-lb knapsack, we have $g(0) = g(1) = g(2) = 0$ and $x(0) = x(1) = x(2) = 0$. Only a Type 2 item fits into a 3-lb knapsack, so we have that $g(3) = 7$ and $x(3) = 2$. Continuing, we find that

$$g(4) = \max \begin{cases} 11 + g(0) = 11^* & \text{(Type 1 item)} \\ 7 + g(1) = 7 & \text{(Type 2 item)} \end{cases}$$

Thus, $g(4) = 11$ and $x(4) = 1$.

$$g(5) = \max \begin{cases} 11 + g(1) = 11 & \text{(Type 1 item)} \\ 7 + g(2) = 7 & \text{(Type 2 item)} \\ 12 + g(0) = 12^* & \text{(Type 3 item)} \end{cases}$$

Thus, $g(5) = 12$ and $x(5) = 3$.

$$g(6) = \max \begin{cases} 11 + g(2) = 11 & \text{(Type 1 item)} \\ 7 + g(3) = 14^* & \text{(Type 2 item)} \\ 12 + g(1) = 12 & \text{(Type 3 item)} \end{cases}$$

Thus, $g(6) = 14$ and $x(6) = 2$.

$$g(7) = \max \begin{cases} 11 + g(3) = 18^* & \text{(Type 1 item)} \\ 7 + g(4) = 18^* & \text{(Type 2 item)} \\ 12 + g(2) = 12 & \text{(Type 3 item)} \end{cases}$$

Thus, $g(7) = 18$ and $x(7) = 1$ or $x(7) = 2$.

$$g(8) = \max \begin{cases} 11 + g(4) = 22^* & \text{(Type 1 item)} \\ 7 + g(5) = 19 & \text{(Type 2 item)} \\ 12 + g(3) = 19 & \text{(Type 3 item)} \end{cases}$$

Thus, $g(8) = 22$ and $x(8) = 1$.

$$g(9) = \max \begin{cases} 11 + g(5) = 23^* & \text{(Type 1 item)} \\ 7 + g(6) = 21 & \text{(Type 2 item)} \\ 12 + g(4) = 23^* & \text{(Type 3 item)} \end{cases}$$

Thus, $g(9) = 23$ and $x(9) = 1$ or $x(9) = 3$.

$$g(10) = \max \begin{cases} 11 + g(6) = 25^* & \text{(Type 1 item)} \\ 7 + g(7) = 25^* & \text{(Type 2 item)} \\ 12 + g(5) = 24 & \text{(Type 3 item)} \end{cases}$$

Thus, $g(10) = 25$ and $x(10) = 1$ or $x(10) = 2$. To fill the knapsack optimally, we begin by putting any $x(10)$ item in the knapsack. Let's arbitrarily choose a Type 1 item. This leaves us with $10 - 4 = 6$ lb to fill, so we now put an $x(10 - 4) = 2$ (Type 2) item in the knapsack. This leaves us with $6 - 3 = 3$ lb to fill, which we do with an $x(6 - 3) = 2$ (Type 2) item. Hence, we may attain the maximum benefit of $g(10) = 25$ by filling the knapsack with two Type 2 items and one Type 1 item.

A Turnpike Theorem

For a knapsack problem, let

c_j = benefit obtained from each type j item

w_j = weight of each type j item

In terms of benefit per unit weight, the best item is the item with the largest value of $\frac{c_j}{w_j}$. Assume there are n types of items that have been ordered, so that

$$\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \cdots \geq \frac{c_n}{w_n}$$

Thus, Type 1 items are the best, Type 2 items are the second best, and so on. Recall from Section 9.5 that it is possible for the optimal solution to a knapsack problem to use none of the best item. For example, the optimal solution to the knapsack problem

$$\begin{aligned} \max z &= 16x_1 + 22x_2 + 12x_3 + 8x_4 \\ \text{s.t.} \quad & 5x_1 + 7x_2 + 5x_3 + 4x_4 \leq 14 \\ & x_i \text{ nonnegative integer} \end{aligned}$$

is $z = 44$, $x_2 = 2$, $x_1 = x_3 = x_4 = 0$, and this solution does not use any of the best (Type 1) item. Assume that

$$\frac{c_1}{w_1} > \frac{c_2}{w_2}$$

Thus, there is a unique best item type. It can be shown that for some number w^*, it is optimal to use at least one Type 1 item if the knapsack is allowed to hold w pounds, where $w \geq w^*$. In Problem 6 at the end of this section, you will show that this result holds for

$$w^* = \frac{c_1 w_1}{c_1 - w_1 \left(\frac{c_2}{w_2}\right)}$$

Thus, for the knapsack problem

$$\begin{aligned} \max z &= 16x_1 + 22x_2 + 12x_3 + 8x_4 \\ \text{s.t.} \quad & 5x_1 + 7x_2 + 5x_3 + 4x_4 \leq w \\ & x_i \text{ nonnegative integer} \end{aligned}$$

at least one Type 1 item will be used if

$$w \geq \frac{16(5)}{16 - 5(\frac{22}{7})} = 280$$

This result can greatly reduce the computation needed to solve a knapsack problem. For example, suppose that $w = 4{,}000$. We know that for $w \geq 280$, the optimal solution will use at least one Type 1 item, so we can conclude that the optimal way to fill a 4,000-lb knapsack will consist of one Type 1 item plus the optimal way to fill a knapsack of $4{,}000 - 5 = 3{,}995$ lb. Repeating this reasoning shows that the optimal way to fill a 4,000-lb knapsack will consist of $\frac{4{,}000-280}{5} = 744$ Type 1 items plus the optimal way to fill a knapsack of 280 lb. This reasoning substantially reduces the computation needed to determine how to fill a 4,000-lb knapsack. (Actually, the 280-lb knapsack will use at least one Type 1 item, so we know that to fill a 4,000-lb knapsack optimally, we can use 745 Type 1 items and then optimally fill a 275-lb knapsack.)

Why is this result referred to as a **turnpike theorem**? Think about taking an automobile trip in which our goal is to minimize the time needed to complete the trip. For a long enough trip, it may be advantageous to go slightly out of our way so that most of the trip will be spent on a turnpike, on which we can travel at the greatest speed. For a short trip, it may not be worth our while to go out of our way to get on the turnpike.

Similarly, in a long (large-weight) knapsack problem, it is always optimal to use some of the best items, but this may not be the case in a short knapsack problem. Turnpike results abound in the dynamic programming literature [see Morton (1979)].

PROBLEMS

Group A

1 J. R. Carrington has $4 million to invest in three oil well sites. The amount of revenue earned from site $i(i = 1, 2, 3)$ depends on the amount of money invested in site i (see Table 10). Assuming that the amount invested in a site must be an exact multiple of $1 million, use dynamic programming to determine an investment policy that will maximize the revenue J. R. will earn from his three oil wells.

2 Use either of the approaches outlined in this section to solve the following knapsack problem:

$$\begin{aligned} \max z &= 5x_1 + 4x_2 + 2x_3 \\ \text{s.t.} \quad 4x_1 &+ 3x_2 + 2x_3 \leq 8 \\ x_1, x_2, x_3 &\geq 0;\ x_1, x_2, x_3 \text{ integer} \end{aligned}$$

3 The knapsack problem of Problem 2 can be viewed as finding the longest route in a particular network.

a Draw the network corresponding to the recursion derived from (7).

b Draw the network corresponding to the recursion derived from (8).

4 The number of crimes in each of a city's three police precincts depends on the number of patrol cars assigned to each precinct (see Table 11). Five patrol cars are available. Use dynamic programming to determine how many patrol cars should be assigned to each precinct.

TABLE 10

Amount Invested ($ Millions)	Revenue ($ Millions)		
	Site 1	Site 2	Site 3
0	4	3	3
1	7	6	7
2	8	10	8
3	9	12	13
4	11	14	15

TABLE 11

	No. of Patrol Cars Assigned to Precinct					
Precinct	0	1	2	3	4	5
1	14	10	7	4	1	0
2	25	19	16	14	12	11
3	20	14	11	8	6	5

5 Use dynamic programming to solve a knapsack problem in which the knapsack can hold up to 13 lb (see Table 12).

Group B

6 Consider a knapsack problem for which

$$\frac{c_1}{w_1} > \frac{c_2}{w_2}$$

Show that if the knapsack can hold w pounds, and $w \geq w^*$, where

$$w^* = \frac{c_1w_1}{\quad}$$

then the optimal solution to the knapsack problem must use at least one Type 1 item.

TABLE 12

Item	Weight (lb)	Benefit
1	3	12
2	5	25
3	7	50

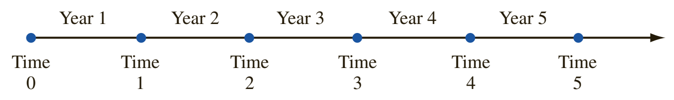
18.5 Equipment-Replacement Problems

Many companies and customers face the problem of determining how long a machine should be utilized before it should be traded in for a new one. Problems of this type are called **equipment-replacement problems** and can often be solved by dynamic programming.

EXAMPLE 7 Equipment Replacement

An auto repair shop always needs to have an engine analyzer available. A new engine an lyzer costs $1,000. The cost m_i of maintaining an engine analyzer during its ith year of operation is as follows: $m_1 = \$60$, $m_2 = \$80$, $m_3 = \$120$. An analyzer may be kept for

FIGURE 8
Time Horizon for Equipment Replacement



1, 2, or 3 years; after i years of use ($i = 1, 2, 3$), it may be traded in for a new one. If an i-year-old engine analyzer is traded in, a salvage value s_i is obtained, where $s_1 = \$800$, $s_2 = \$600$, and $s_3 = \$500$. Given that a new machine must be purchased now (time 0; see Figure 8), the shop wants to determine a replacement and trade-in policy that minimizes net costs = (maintenance costs) + (replacement costs) – (salvage value received) during the next 5 years.

Solution We note that after a new machine is purchased, the firm must decide when the newly purchased machine should be traded in for a new one. With this in mind, we define $g(t)$ to be the minimum net cost incurred from time t until time 5 (including the purchase cost and salvage value for the newly purchased machine) given that a new machine has been purchased at time t. We also define c_{tx} to be the net cost (including purchase cost and salvage value) of purchasing a machine at time t and operating it until time x. Then the appropriate recursion is

$$g(t) = \min_{x}\{c_{tx} + g(x)\} \qquad (t = 0, 1, 2, 3, 4) \tag{9}$$

where x must satisfy the inequalities $t + 1 \leq x \leq t + 3$ and $x \leq 5$. Because the problem is over at time 5, no cost is incurred from time 5 onward, so we may write $g(5) = 0$.

To justify (9), note that after a new machine is purchased at time t, we must decide when to replace the machine. Let x be the time at which the replacement occurs. The replacement must be after time t but within 3 years of time t. This explains the restriction that $t + 1 \leq x \leq t + 3$. Since the problem ends at time 5, we must also have $x \leq 5$. If we choose to replace the machine at time x, then what will be the cost from time t to time 5? Simply the sum of the cost incurred from the purchase of the machine to the sale of the machine at time x (which is by definition c_{tx}) and the total cost incurred from time x to time 5 (given that a new machine has just been purchased at time x). By the principle of optimality, the latter cost is, of course, $g(x)$. Hence, if we keep the machine that was purchased at time t until time x, then from time t to time 5, we incur a cost of $c_{tx} + g(x)$. Thus, x should be chosen to minimize this sum, and this is exactly what (9) does. We have assumed that maintenance costs, salvage value, and purchase price remain unchanged over time, so each c_{tx} will depend only on how long the machine is kept; that is, each c_{tx} depends only on $x - t$. More specifically,

$$c_{tx} = \$1{,}000 + m_1 + \cdots + m_{x-t} - s_{x-t}$$

This yields

$$c_{01} = c_{12} = c_{23} = c_{34} = c_{45} = 1{,}000 + 60 - 800 = \$260$$
$$c_{02} = c_{13} = c_{24} = c_{35} = 1{,}000 + 60 + 80 - 600 = \$540$$
$$c_{03} = c_{14} = c_{25} = 1{,}000 + 60 + 80 + 120 - 500 = \$760$$

We begin by computing $g(4)$ and work backward until we have computed $g(0)$. Then we use our knowledge of the values of x attaining $g(0)$, $g(1)$, $g(2)$, $g(3)$, and $g(4)$ to determine the optimal replacement strategy. The calculations follow.

At time 4, there is only one sensible decision (keep the machine until time 5 and sell it for its salvage value), so we find

$$g(4) = c_{45} + g(5) = 260 + 0 = \$260^*$$

Thus, if a new machine is purchased at time 4, it should be traded in at time 5.

If a new machine is purchased at time 3, we keep it until time 4 or time 5. Hence,

$$g(3) = \min \begin{cases} c_{34} + g(4) = 260 + 260 = \$520* & \text{(Trade at time 4)} \\ c_{35} + g(5) = 540 + 0 = \$540 & \text{(Trade at time 5)} \end{cases}$$

Thus, if a new machine is purchased at time 3, we should trade it in at time 4.

If a new machine is purchased at time 2, we trade it in at time 3, time 4, or time 5. This yields

$$g(2) = \min \begin{cases} c_{23} + g(3) = 260 + 520 = \$780 & \text{(Trade at time 3)} \\ c_{24} + g(4) = 540 + 260 = \$800 & \text{(Trade at time 4)} \\ c_{25} + g(5) = \$760* & \text{(Trade at time 5)} \end{cases}$$

Thus, if we purchase a new machine at time 2, we should keep it until time 5 and then trade it in.

If a new machine is purchased at time 1, we trade it in at time 2, time 3, or time 4. Then

$$g(1) = \min \begin{cases} c_{12} + g(2) = 260 + 760 = \$1{,}020* & \text{(Trade at time 2)} \\ c_{13} + g(3) = 540 + 520 = \$1{,}060 & \text{(Trade at time 3)} \\ c_{14} + g(4) = 760 + 260 = \$1{,}020* & \text{(Trade at time 4)} \end{cases}$$

Thus, if a new machine is purchased at time 1, it should be traded in at time 2 or time 4.

The new machine that was purchased at time 0 may be traded in at time 1, time 2, or time 3. Thus,

$$g(0) = \min \begin{cases} c_{01} + g(1) = 260 + 1{,}020 = \$1{,}280* & \text{(Trade at time 1)} \\ c_{02} + g(2) = 540 + 760 = \$1{,}300 & \text{(Trade at time 2)} \\ c_{03} + g(3) = 760 + 520 = \$1{,}280* & \text{(Trade at time 3)} \end{cases}$$

Thus, the new machine purchased at time 0 should be replaced at time 1 or time 3. Let's arbitrarily choose to replace the time 0 machine at time 1. Then the new time 1 machine may be traded in at time 2 or time 4. Again we make an arbitrary choice and replace the time 1 machine at time 2. Then the time 2 machine should be kept until time 5, when it is sold for salvage value. With this replacement policy, we will incur a net cost of $g(0) =$ \$1,280. The reader should verify that the following replacement policies are also optimal: (1) trading in at times 1, 4, and 5 and (2) trading in at times 3, 4, and 5.

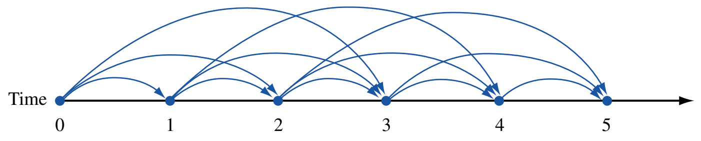
We have assumed that all costs remain stationary over time. This assumption was made solely to simplify the computation of the c_{tx}'s. If we had relaxed the assumption of stationary costs, then the only complication would have been that the c_{tx}'s would have been messier to compute. We also note that if a short planning horizon is used, the optimal replacement policy may be extremely sensitive to the length of the planning horizon. Thus, more meaningful results can be obtained by using a longer planning horizon.

An equipment-replacement model was actually used by Phillips Petroleum to reduce costs associated with maintaining the company's stock of trucks (see Waddell (1983)).

Network Representation of Equipment-Replacement Problem

The reader should verify that our solution to Example 7 was equivalent to finding the shortest path from node 0 to node 5 in the network in Figure 9. The length of the arc joining nodes i and j is c_{ij}.

FIGURE 9
Network Representation of Equipment Replacement



An Alternative Recursion

There is another dynamic programming formulation of the equipment-replacement model. If we define the stage to be the time t and the state at any stage to be the age of the engine analyzer at time t, then an alternative dynamic programming recursion can be developed. Define $f_t(x)$ to be the minimum cost incurred from time t to time 5, given that at time t the shop has an x-year-old analyzer. The problem is over at time 5, so we sell the machine at time 5 and receive $-s_x$. Then $f_5(x) = -s_x$, and for $t = 0, 1, 2, 3, 4,$

$$f_t(3) = -500 + 1{,}000 + 60 + f_{t+1}(1) \qquad \text{(Trade)} \qquad \textbf{(10)}$$

$$f_t(2) = \min\begin{cases} -600 + 1{,}000 + 60 + f_{t+1}(1) & \text{(Trade)} \\ 120 + f_{t+1}(3) & \text{(Keep)} \end{cases} \qquad \textbf{(10.1)}$$

$$f_t(1) = \min\begin{cases} -800 + 1{,}000 + 60 + f_{t+1}(1) & \text{(Trade)} \\ 80 + f_{t+1}(2) & \text{(Keep)} \end{cases} \qquad \textbf{(10.2)}$$

$$f_0(0) = 1{,}000 + 60 + f_1(1) \qquad \text{(Keep)} \qquad \textbf{(10.3)}$$

The rationale behind Equations (10)–(10.3) is that if we have a 1- or 2-year-old analyzer, then we must decide between replacing the machine or keeping it another year. In (10.1) and (10.2), we compare the costs of these two options. For any option, the total cost from t until time 5 is the sum of the cost during the current year plus costs from time $t + 1$ to time 5. If we have a 3-year-old analyzer, then we must replace it, so there is no choice. The way we have defined the state means that it is only possible to be in state 0 at time 0. In this case, we must keep the analyzer for the first year (incurring a cost of $1,060). From this point on, a total cost of $f_1(1)$ is incurred. Thus, (10.3) follows. Since we know that $f_5(1) = -800$, $f_5(2) = -600$, and $f_5(3) = -500$, we can immediately compute all the $f_4(\cdot)$'s. Then we can compute the $f_3(\cdot)$'s. We continue in this fashion until $f_0(0)$ is determined (remember that we begin with a new machine). Then we follow our usual method for determining an optimal policy. That is, if $f_0(0)$ is attained by keeping the machine, then we keep the machine for a year and then, during year 1, we choose the action that attains $f_1(1)$. Continuing in this fashion, we can determine for each time whether or not the machine should be replaced. (See Problem 1 below.)

PROBLEMS

Group A

1 Use Equations (10)–(10.3) to determine an optimal replacement policy for the engine analyzer example.

2 Suppose that a new car costs $10,000 and that the annual operating cost and resale value of the car are as shown in Table 13. If I have a new car now, determine a replacement policy that minimizes the net cost of owning and operating a car for the next six years.

3 It costs $40 to buy a telephone from a department store. The estimated maintenance cost for each year of operation is shown in Table 14. (I can keep a telephone for at most five years.) I have just purchased a new telephone, and my old telephone has no salvage value. Determine how to minimize the total cost of purchasing and operating a telephone for the next six years.

TABLE 13

Age of Car (Years)	Resale Value ($)	Operating Cost ($)
1	7,000	300 (year 1)
2	6,000	500 (year 2)
3	4,000	800 (year 3)
4	3,000	1,200 (year 4)
5	2,000	1,600 (year 5)
6	1,000	2,200 (year 6)

TABLE 14

Year	Maintenance Cost ($)
1	20
2	30
3	40
4	60
5	70

18.6 Formulating Dynamic Programming Recursions

In many dynamic programming problems (such as the inventory and shortest path examples), a given stage simply consists of all the possible states that the system can occupy at that stage. If this is the case, then the dynamic programming recursion (for a min problem) can often be written in the following form:

$$f_t(i) = \min\{(\text{cost during stage } t) + f_{t+1}\ (\text{new state at stage } t + 1)\} \tag{11}$$

where the minimum in (11) is over all decisions that are allowable, or feasible, when the state at stage t is i. In (11), $f_t(i)$ is the minimum cost incurred from stage t to the end of the problem (say, the problem ends after stage T), given that at stage t the state is i.

Equation (11) reflects the fact that the minimum cost incurred from stage t to the end of the problem must be attained by choosing at stage t an allowable decision that minimizes the sum of the costs incurred during the current stage (stage t) plus the minimum cost that can be incurred from stage $t + 1$ to the end of the problem. Correct formulation of a recursion of the form (11) requires that we identify three important aspects of the problem:

Aspect 1 *The set of decisions that is allowable, or feasible, for the given state and stage.* Often, the set of feasible decisions depends on both t and i. For instance, in the inventory example of Section 18.3, let

$$d_t = \text{demand during month } t$$
$$i_t = \text{inventory at beginning of month } t$$

In this case, the set of allowable month t decisions (let x_t represent an allowable production level) consists of the members of $\{0, 1, 2, 3, 4, 5\}$ that satisfy $0 \le (i_t + x_t - d_t) \le 4$. Note how the set of allowable decisions at time t depends on the stage t and the state at time t, which is i_t.

Aspect 2 *We must specify how the cost during the current time period (stage t) depends on the value of t, the current state, and the decision chosen at stage t.* For instance, in the inventory example of Section 18.3, suppose a production level x_t is chosen during month t. Then the cost during month t is given by $c(x_t) + (\frac{1}{2})(i_t + x_t - d_t)$.

Aspect 3 *We must specify how the state at stage t + 1 depends on the value of t, the state at stage t, and the decision chosen at stage t.* Again referring to the inventory example, the month $t + 1$ state is $i_t + x_t - d_t$.

If you have properly identified the state, stage, and decision, then aspects 1–3 shouldn't be too hard to handle. A word of caution, however: Not all recursions are of the form (11). For instance, our first equipment-replacement recursion skipped over time $t + 1$.

This often occurs when the stage alone supplies sufficient information to make an optimal decision. We now work through several examples that illustrate the art of formulating dynamic programming recursions.

EXAMPLE 8 A Fishery

The owner of a lake must decide how many bass to catch and sell each year. If she sells x bass during year t, then a revenue $r(x)$ is earned. The cost of catching x bass during a year is a function $c(x, b)$ of the number of bass caught during the year and of b, the number of bass in the lake at the beginning of the year. Of course, bass do reproduce. To model this, we assume that the number of bass in the lake at the beginning of a year is 20% more than the number of bass left in the lake at the end of the previous year. Assume that there are 10,000 bass in the lake at the beginning of the first year. Develop a dynamic programming recursion that can be used to maximize the owner's net profits over a T-year horizon.

Solution In problems where decisions must be made at several points in time, there is often a trade-off of current benefits against future benefits. For example, we could catch many bass early in the problem, but then the lake would be depleted in later years, and there would be very few bass to catch. On the other hand, if we catch very few bass now, we won't make much money early, but we can make a lot of money near the end of the horizon. In intertemporal optimization problems, dynamic programming is often used to analyze these complex trade-offs.

At the beginning of year T, the owner of the lake need not worry about the effect that the capture of bass will have on the future population of the lake. (At time T, there is no future!) So at the beginning of year T, the problem is relatively easy to solve. For this reason, we let time be the stage. At each stage, the owner of the lake must decide how many bass to catch. We define x_t to be the number of bass caught during year t. To determine an optimal value of x_t, the owner of the lake need only know the number of bass (call it b_t) in the lake at the beginning of year t. Therefore, the state at the beginning of year t is b_t.

We define $f_t(b_t)$ to be the maximum net profit that can be earned from bass caught during years $t, t + 1, \ldots, T$ given that b_t bass are in the lake at the beginning of year t. We may now dispose of aspects 1–3 of the recursion.

Aspect 1 What are the allowable decisions? During any year, we can't catch more bass than there are in the lake. Thus, in each state and for all t, $0 \leq x_t \leq b_t$ must hold.

Aspect 2 What is the net profit earned during year t? If x_t bass are caught during a year that begins with b_t bass in the lake, then the net profit is $r(x_t) - c(x_t, b_t)$.

Aspect 3 What will be the state during year $t + 1$? At the end of year t, there will be $b_t - x_t$ bass in the lake. By the beginning of year $t + 1$, these bass will have multiplied by 20%. This implies that at the beginning of year $t + 1$, $1.2(b_t - x_t)$ bass will be in the lake. Thus, the year $t + 1$ state will be $1.2(b_t - x_t)$.

We can now use (11) to develop the appropriate recursion. After year T, there are no future profits to consider, so

$$f_T(b_T) = \max_{x_T}\{r_T(x_T) - c(x_T, b_T)\}$$

where $0 \leq x_T \leq b_T$. Applying (11), we obtain

$$f_t(b_t) = \max\{r(x_t) - c(x_t, b_t) + f_{t+1}[1.2(b_t - x_t)]\} \tag{12}$$

where $0 \leq x_t \leq b_t$. To begin the computations, we first determine $f_T(b_T)$ for all values of b_T that might occur [b_T could be up to $10{,}000(1.2)^{T-1}$; why?]. Then we use (12) to work backward until $f_1(10{,}000)$ has been computed. Then, to determine an optimal fishing policy, we begin by choosing x_1 to be any value attaining the maximum in the (12) equation for $f_1(10{,}000)$. Then year 2 will begin with $1.2(10{,}000 - x_1)$ bass in the lake. This means that x_2 should be chosen to be any value attaining the maximum in the (12) equation for $f_2(1.2(10{,}000 - x_1))$. Continue in this fashion until the optimal values of $x_3, x_4, \ldots, x_T$ have been determined.

Incorporating the Time Value of Money into Dynamic Programming Formulations

A weakness of the current formulation is that profits received during later years are weighted the same as profits received during earlier years. As mentioned in the discussion of discounting (in Chapter 3), later profits should be weighted less than earlier profits. Suppose that for some $\beta < 1$, \$1 received at the beginning of year $t + 1$ is equivalent to β dollars received at the beginning of year t. We can incorporate this idea into the dynamic programming recursion by replacing (12) with

$$f_t(b_t) = \max_{x_t} \{r(x_t) - c(x_t, b_t) + \beta f_{t+1}[1.2(b_t - x_t)]\} \tag{12'}$$

where $0 \le x_t \le b_t$. Then we redefine $f_t(b_t)$ to be the maximum net profit *(in year t dollars)* that can be earned during years $t, t + 1, \ldots, T$. Since f_{t+1} is measured in year $t + 1$ dollars, multiplying it by β converts $f_{t+1}(\cdot)$ to year t dollars, which is just what we want. In Example 8, once we have worked backward and determined $f_1(10{,}000)$, an optimal fishing policy is found by using the same method that was previously described. This approach can be used to account for the time value of money in any dynamic programming formulation.

EXAMPLE 9 **Power Plant**

An electric power utility forecasts that r_t kilowatt-hours (kwh) of generating capacity will be needed during year t (the current year is year 1). Each year, the utility must decide by how much generating capacity should be expanded. It costs $c_t(x)$ dollars to increase generating capacity by x kwh during year t. It may be desirable to reduce capacity, so x need not be nonnegative. During each year, 10% of the old generating capacity becomes obsolete and unusable (capacity does not become obsolete during its first year of operation). It costs the utility $m_t(i)$ dollars to maintain i units of capacity during year t. At the beginning of year 1, 100,000 kwh of generating capacity are available. Formulate a dynamic programming recursion that will enable the utility to minimize the total cost of meeting power requirements for the next T years.

Solution Again, we let time be the stage. At the beginning of year t, the utility must determine the amount of capacity (call it x_t) to add during year t. To choose x_t properly, all the utility needs to know is the amount of available capacity at the beginning of year t (call it i_t). Hence, we define the state at the beginning of year t to be the current capacity level. We may now dispose of aspects 1–3 of the formulation.

Aspect 1 What values of x_t are feasible? To meet year t's requirement of r_t, we must have $i_t + x_t \ge r_t$, or $x_t \ge r_t - i_t$. So the feasible x_t's are those values of x_t satisfying $x_t \ge r_t - i_t$.

Aspect 2 What cost is incurred during year t? If x_t kwh are added during a year that begins with i_t kwh of available capacity, then during year t, a cost $c_t(x_t) + m_t(i_t + x_t)$ is incurred.

Aspect 3 What will be the state at the beginning of year $t + 1$? At the beginning of year $t + 1$, the utility will have $0.9i_t$ kwh of old capacity plus the x_t kwh that have been added during year t. Thus, the state at the beginning of year $t + 1$ will be $0.9i_t + x_t$.

We can now use (11) to develop the appropriate recursion. Define $f_t(i_t)$ to be the minimum cost incurred by the utility during years $t, t+1, \ldots, T$, given that i_t kwh of capacity are available at the beginning of year t. At the beginning of year T, there are no future costs to consider, so

$$f_T(i_T) = \min_{x_T} \{c_T(x_T) + m_T(i_T + x_T)\} \tag{13}$$

where x_T must satisfy $x_T \geq r_T - i_T$. For $t < T$,

$$f_t(i_t) = \min_{x_T} \{c_t(x_t) + m_t(i_t + x_t) + f_{t+1}(0.9i_t + x_t)\} \tag{14}$$

where x_t must satisfy $x_t \geq r_t - i_t$. If the utility does not start with any excess capacity, then we can safely assume that the capacity level would never exceed $r_{\text{MAX}} = \max\limits_{t=1,\,2,\ldots,\,T} \{r_t\}$. This means that we need consider only states $0, 1, 2, \ldots, r_{\text{MAX}}$. To begin computations, we use (13) to compute $f_T(0), f_T(1), \ldots, f_T(r_{\text{MAX}})$. Then we use (14) to work backward until $f_1(100{,}000)$ has been determined. To determine the optimal amount of capacity that should be added during each year, proceed as follows. During year 1, add an amount of capacity x_1 that attains the minimum in the (14) equation for $f_1(100{,}000)$. Then the utility will begin year 2 with $90{,}000 + x_1$ kwh of capacity. Then, during year 2, x_2 kwh of capacity should be added, where x_2 attains the minimum in the (14) equation for $f_2(90{,}000 + x_1)$. Continue in this fashion until the optimal value of x_T has been determined.

EXAMPLE 10 Wheat Sale

Farmer Jones now possesses \$5,000 in cash and 1,000 bushels of wheat. During month t, the price of wheat is p_t. During each month, he must decide how many bushels of wheat to buy (or sell). There are three restrictions on each month's wheat transactions: (1) During any month, the amount of money spent on wheat cannot exceed the cash on hand at the beginning of the month; (2) during any month, he cannot sell more wheat than he has at the beginning of the month; and (3) because of limited warehouse capacity, the ending inventory of wheat for each month cannot exceed 1,000 bushels.

Show how dynamic programming can be used to maximize the amount of cash that farmer Jones has on hand at the end of six months.

Solution Again, we let time be the stage. At the beginning of month t (the present is the beginning of month 1), farmer Jones must decide by how much to change the amount of wheat on hand. We define Δw_t to be the change in farmer Jones's wheat position during month t: $\Delta w_t \geq 0$ corresponds to a month t wheat purchase, and $\Delta w_t \leq 0$ corresponds to a month t sale of wheat. To determine an optimal value for Δw_t, we must know two things: the amount of wheat on hand at the beginning of month t (call it w_t) and the cash on hand at the beginning of month t, (call this c_t). We define $f_t(c_t, w_t)$ to be the maximum cash that farmer Jones can obtain at the end of month 6, given that farmer Jones has c_t dollars and w_t bushels of wheat at the beginning of month t. We now discuss aspects 1–3 of the formulation.

Aspect 1 What are the allowable decisions? If the state at time t is (c_t, w_t), then restrictions 1–3 limit Δw_t in the following manner:

$$p_t(\Delta w_t) \leq c_t \qquad \text{or} \qquad \Delta w_t \leq \frac{c_t}{p_t}$$

ensures that we won't run out of money at the end of month t. The inequality $\Delta w_t \geq -w_t$ ensures that during month t, we will not sell more wheat than we had at the beginning of month t; and $w_t + \Delta w_t \leq 1{,}000$, or $\Delta w_t \leq 1{,}000 - w_t$, ensures that we will end month t with at most 1,000 bushels of wheat. Putting these three restrictions together, we see that

$$-w_t \leq \Delta w_t \leq \min\left\{\frac{c_t}{p_t}, 1{,}000 - w_t\right\}$$

will ensure that restrictions 1–3 are satisfied during month t.

Aspect 2 Since farmer Jones wants to maximize his cash on hand at the end of month 6, no benefit is earned during months 1 through 5. In effect, during months 1–5, we are doing bookkeeping to keep track of farmer Jones's position. Then, during month 6, we turn all of farmer Jones's assets into cash.

Aspect 3 If the current state is (c_t, w_t) and farmer Jones changes his month t wheat position by an amount Δw_t, what will be the new state at the beginning of month $t + 1$? Cash on hand will increase by $-(\Delta w_t)p_t$, and farmer Jones's wheat position will increase by Δw_t. Hence, the month $t + 1$ state will be $[c_t - (\Delta w_t)p_t, w_t + \Delta w_t]$.

We may now use (11) to develop the appropriate recursion. To maximize his cash position at the end of month 6, farmer Jones should convert his month 6 wheat into cash by selling all of it. This means that $\Delta w_6 = -w_6$. This leads to the following relation:

$$f_6(c_6, w_6) = c_6 + w_6 p_6 \tag{15}$$

Using (11), we obtain for $t < 6$

$$f_t(c_t, w_t) = \max_{\Delta w_t} \{0 + f_{t+1}[c_t - (\Delta w_t)p_t, w_t + \Delta w_t]\} \tag{16}$$

where Δw_t must satisfy

$$-w_t \leq \Delta w_t \leq \min\left\{\frac{c_t}{p_t}, 1{,}000 - w_t\right\}$$

We begin our calculations by determining $f_6(c_6, w_6)$ for all states that can possibly occur during month 6. Then we use (16) to work backward until $f_1(5{,}000, 1{,}000)$ has been computed. Next, farmer Jones should choose Δw_1 to attain the maximum value in the (16) equation for $f_1(5{,}000, 1{,}000)$, and a month 2 state of $[5{,}000 - p_1(\Delta w_1), 1{,}000 + \Delta w_1]$ will ensue. Farmer Jones should next choose Δw_2 to attain the maximum value in the (16) equation for $f_2[5{,}000 - p_1(\Delta w_1), 1{,}000 + \Delta w_1]$. We continue in this manner until the optimal value of Δw_6 has been determined.

EXAMPLE 11 Refinery Capacity

Sunco Oil needs to build enough refinery capacity to refine 5,000 barrels of oil per day and 10,000 barrels of gasoline per day. Sunco can build refinery capacity at four locations. The cost of building a refinery at site t that has the capacity to refine x barrels of oil per day and y barrels of gasoline per day is $c_t(x, y)$. Use dynamic programming to determine how much capacity should be located at each site.

Solution If Sunco had only one possible refinery site, then the problem would be easy to solve. Sunco could solve a problem in which there were two possible refinery sites, and finally, a problem in which there were four refinery sites. For this reason, we let the stage represent the number of available oil sites. At any stage, Sunco must determine how much oil and gas capacity should be built at the given site. To do this, the company must know how much refinery capacity of each type must be built at the available sites. We now define $f_t(o_t, g_t)$ to be the minimum cost of building o_t barrels per day of oil refinery capacity and g_t barrels per day of gasoline refinery capacity at sites $t, t + 1, \ldots, 4$.

To determine $f_4(o_4, g_4)$, note that if only site 4 is available, Sunco must build a refinery at site 4 with o_4 barrels of oil capacity and g_4 barrels of gasoline capacity. This implies that $f_4(o_4, g_4) = c_4(o_4, g_4)$. For $t = 1, 2, 3$, we can determine $f_t(o_t, g_t)$ by noting that if we build a refinery at site t that can refine x_t barrels of oil per day and y_t barrels of gasoline per day, then we incur a cost of $c_t(x_t, y_t)$ at site t. Then we will need to build a total oil refinery capacity of $o_t - x_t$ and a gas refinery capacity of $g_t - y_t$ at sites $t + 1$, $t + 2, \ldots, 4$. By the principle of optimality, the cost of doing this will be $f_{t+1}(o_t - x_t, g_t - y)$. Since $0 \le x_t \le o_t$ and $0 \le y_t \le g_t$ must hold, we obtain the following recursion:

$$f_t(o_t, g_t) = \min \{c_t(o_t, g_t) + f_{t+1}(o_t - x_t, g_t - y_t)\} \quad (17)$$

where $0 \le x_t \le o_t$ and $0 \le y_t \le g_t$. As usual, we work backward until $f_1(5{,}000, 10{,}000)$ has been determined. Then Sunco chooses x_1 and y_1 to attain the minimum in the (17) equation for f_1 (5,000, 10,000). Then Sunco should choose x_2 and y_2 that attain the minimum in the (17) equation for $f_2(5{,}000 - x_1, 10{,}000 - y_1)$. Sunco continues in this fashion until optimal values of x_4 and y_4 are determined.

EXAMPLE 12 Traveling Salesperson

The traveling salesperson problem (see Section 9.6) can be solved by using dynamic programming. As an example, we solve the following traveling salesperson problem: It's the last weekend of the 2004 election campaign, and candidate Walter Glenn is in New York City. Before election day, Walter must visit Miami, Dallas, and Chicago and then return to his New York City headquarters. Walter wants to minimize the total distance he must travel. In what order should he visit the cities? The distances in miles between the four cities are given in Table 15.

Solution We know that Walter must visit each city exactly once, the last city he visits must be New York, and his tour originates in New York. When Walter has only one city left to visit, his problem is trivial: simply go from his current location to New York. Then we can work backward to a problem in which he is in some city and has only two cities left to visit, and finally we can find the shortest tour that originates in New York and has four cities left to visit. We therefore let the stage be indexed by the number of cities that Walter has already visited. At any stage, to determine which city should next be visited, we need to know two things: Walter's current location and the cities he has already visited. The state at any stage consists of the last city visited and the set of cities that have already been visited. We define $f_t(i, S)$ to be the minimum distance that must be traveled to complete a tour if the $t - 1$ cities in the set S have been visited and city i was the last city visited. We let c_{ij} be the distance between cities i and j.

Stage 4 Computations

We note that, at stage 4, it must be the case that $S = \{2, 3, 4\}$ (why?), and the only possible states are $(2, \{2, 3, 4\})$, $(3, \{2, 3, 4\})$, and $(4, \{2, 3, 4\})$. In stage 4, we must go from the current location to New York. This observation yields

TABLE 15
Distances for a Traveling Salesperson

	City			
	New York	Miami	Dallas	Chicago
1 New York	—	1,334	1,559	809
2 Miami	1,334	—	1,343	1,397
3 Dallas	1,559	1,343	—	921
4 Chicago	809	1,397	921	—

$$f_4(2, \{2, 3, 4\}) = c_{21} = 1{,}334^* \qquad \text{(Go from city 2 to city 1)}$$
$$f_4(3, \{2, 3, 4\}) = c_{31} = 1{,}559^* \qquad \text{(Go from city 3 to city 1)}$$
$$f_4(4, \{2, 3, 4\}) = c_{41} = 809^* \qquad \text{(Go from city 4 to city 1)}$$

Stage 3 Computations

Working backward to stage 3, we write

$$f_3(i, S) = \min_{\substack{j \notin S \\ \text{and } j \neq 1}} \{c_{ij} + f_4[j, S \cup \{j\}]\} \tag{18}$$

This result follows, because if Walter is now at city i and he travels to city j, he travels a distance c_{ij}. Then he is at stage 4, has last visited city j, and has visited the cities in $S \cup \{j\}$. Hence, the length of the rest of his tour must be $f_4(j, S \cup \{j\})$. To use (18), note that at stage 3, Walter must have visited $\{2, 3\}$, $\{2, 4\}$, or $\{3, 4\}$ and must next visit the nonmember of S that is not equal to 1. We can use (18) to determine $f_3(\cdot)$ for all possible states:

$$f_3(2, \{2, 3\}) = c_{24} + f_4(4, \{2, 3, 4\}) = 1{,}397 + 809 = 2{,}206^* \qquad \text{(Go from 2 to 4)}$$
$$f_3(3, \{2, 3\}) = c_{34} + f_4(4, \{2, 3, 4\}) = 921 + 809 = 1{,}730^* \qquad \text{(Go from 3 to 4)}$$
$$f_3(2, \{2, 4\}) = c_{23} + f_4(3, \{2, 3, 4\}) = 1{,}343 + 1{,}559 = 2{,}902^* \qquad \text{(Go from 2 to 3)}$$
$$f_3(4, \{2, 4\}) = c_{43} + f_4(3, \{2, 3, 4\}) = 921 + 1{,}559 = 2{,}480^* \qquad \text{(Go from 4 to 3)}$$
$$f_3(3, \{3, 4\}) = c_{32} + f_4(2, \{2, 3, 4\}) = 1{,}343 + 1{,}334 = 2{,}677^* \qquad \text{(Go from 3 to 2)}$$
$$f_3(4, \{3, 4\}) = c_{42} + f_4(2, \{2, 3, 4\}) = 1{,}397 + 1{,}334 = 2{,}731^* \qquad \text{(Go from 4 to 2)}$$

In general, we write, for $t = 1, 2, 3$,

$$f_t(i, S) = \min_{\substack{j \notin S \\ \text{and } j \neq 1}} \{c_{ij} + f_{t+1}[j, S \cup \{j\}]\} \tag{19}$$

This result follows, because if Walter is at present in city i and he next visits city j, then he travels a distance c_{ij}. The remainder of his tour will originate from city j, and he will have visited the cities in $S \cup \{j\}$. Hence, the length of the remainder of his tour must be $f_{t+1}(j, S \cup \{j\})$. Equation (19) now follows.

Stage 2 Computations

At stage 2, Walter has visited only one city, so the only possible states are (2, {2}), (3, {3}), and (4, {4}). Applying (19), we obtain

$$f_2(2, \{2\}) = \min \begin{cases} c_{23} + f_3(3, \{2, 3\}) = 1{,}343 + 1{,}730 = 3{,}073^* \\ \text{(Go from 2 to 3)} \\ c_{24} + f_3(4, \{2, 4\}) = 1{,}397 + 2{,}480 = 3{,}877 \\ \text{(Go from 2 to 4)} \end{cases}$$

$$f_2(3, \{3\}) = \min \begin{cases} c_{34} + f_3(4, \{3, 4\}) = 921 + 2{,}731 = 3{,}652 \\ \text{(Go from 3 to 4)} \\ c_{32} + f_3(2, \{2, 3\}) = 1{,}343 + 2{,}206 = 3{,}549^* \\ \text{(Go from 3 to 2)} \end{cases}$$

$$f_2(4, \{4\}) = \min \begin{cases} c_{42} + f_3(2, \{2, 4\}) = 1{,}397 + 2{,}902 = 4{,}299 \\ \text{(Go from 4 to 2)} \\ c_{43} + f_3(3, \{3, 4\}) = 921 + 2{,}677 = 3{,}598^* \\ \text{(Go from 4 to 3)} \end{cases}$$

Stage 1 Computations

Finally, we are back to stage 1 (where no cities have been visited). Since Walter is currently in New York and has visited no cities, the stage 1 state must be $f_1(1, \{\cdot\})$. Applying (19),

$$f_1(1, \{\cdot\}) = \min \begin{cases} c_{12} + f_2(2, \{2\}) = 1{,}334 + 3{,}073 = 4{,}407^* \\ \text{(Go from 1 to 2)} \\ c_{13} + f_2(3, \{3\}) = 1{,}559 + 3{,}549 = 5{,}108 \\ \text{(Go from 1 to 3)} \\ c_{14} + f_2(4, \{4\}) = 809 + 3{,}598 = 4{,}407^* \\ \text{(Go from 1 to 4)} \end{cases}$$

So from city 1 (New York), Walter may go to city 2 (Miami) or city 4 (Chicago). We arbitrarily have him choose to go to city 4. Then he must choose to visit the city that attains $f_2(4, \{4\})$, which requires that he next visit city 3 (Dallas). Then he must visit the city attaining $f_3(3, \{3, 4\})$, which requires that he next visit city 2 (Miami). Then Walter must visit the city attaining $f_4(2, \{2, 3, 4\})$, which means, of course, that he must next visit city 1 (New York). The optimal tour (1–4–3–2–1, or New York–Chicago–Dallas–Miami–New York) is now complete. The length of this tour is $f_1(1, \{\cdot\}) = 4{,}407$. As a check, note that

New York to Chicago distance = 809 miles

Chicago to Dallas distance = 921 miles

Dallas to Miami distance = 1,343 miles

Miami to New York distance = 1,334 miles

so the total distance that Walter travels is $809 + 921 + 1{,}343 + 1{,}334 = 4{,}407$ miles. Of course, if we had first sent him to city 2, we would have obtained another optimal tour (1–2–3–4–1) that would simply be a reversal of the original optimal tour.

Computational Difficulties in Using Dynamic Programming

For traveling salesperson problems that are large, the state space becomes very large, and the branch-and-bound approach outlined in Chapter 9 (along with other branch-and-bound approaches) is much more efficient than the dynamic programming approach outlined here. For example, for a 30-city problem, suppose we are at stage 16 (this means that 15 cities have been visited). Then it can be shown that there are more than 1 billion possible states. This brings up a problem that limits the practical application of dynamic programming. In many problems, *the state space becomes so large that excessive computational time is required to solve the problem by dynamic programming.* For instance, in Example 8, suppose that $T = 20$. It is possible that if no bass were caught during the first 20 years, then the lake might contain $10{,}000(1.2)^{20} = 383{,}376$ bass at the beginning of year 21. If we view this example as a network in which we need to find the longest route from the node (1, 10,000) (representing year 1 and 10,000 bass in the lake) to some stage 21 node, then stage 21 would have 383,377 nodes. Even a powerful computer would have difficulty solving this problem. Techniques to make problems with large state spaces computationally tractable are discussed in Bersetkas (1987) and Denardo (1982).

Nonadditive Recursions

The last two examples in this section differ from the previous ones in that the recursion does not represent $f_t(i)$ as the sum of the cost (or reward) incurred during the current period and future costs (or rewards) incurred during future periods.

EXAMPLE 13 Minimax Shortest Route

Joe Cougar needs to drive from city 1 to city 10. He is no longer interested in minimizing the length of his trip, but he is interested in minimizing the maximum altitude above sea level that he will encounter during his drive. To get from city 1 to city 10, he must follow a path in Figure 10. The length c_{ij} of the arc connecting city i and city j represents the maximum altitude (in thousands of feet above sea level) encountered when driving from city i to city j. Use dynamic programming to determine how Joe should proceed from city 1 to city 10.

Solution To solve this problem by dynamic programming, note that for a trip that begins in city i and goes through stages $t, t + 1, \ldots, 5$, the maximum altitude that Joe encounters will be the maximum of the following two quantities: (1) the maximum altitude encountered on stages $t + 1, t + 2, \ldots, 5$ or (2) the altitude encountered when traversing the arc that begins in stage t. Of course, if we are in a stage 4 state, quantity 1 does not exist.

After defining $f_t(i)$ as the smallest maximum altitude that Joe can encounter in a trip from city i in stage t to city 10, this reasoning leads us to the following recursion:

$$
\begin{aligned}
f_4(i) &= c_{i,10} \\
f_t(i) &= \min_j \{\max[c_{ij}, f_{t+1}(j)]\} \qquad (t = 1, 2, 3)
\end{aligned}
\tag{20}
$$

where j may be any city such that there is an arc connecting city i and city j.

We first compute $f_4(7)$, $f_4(8)$, and $f_4(9)$ and then use (20) to work backward until $f_1(1)$ has been computed. We obtain the following results:

$$
\begin{aligned}
f_4(7) &= 13^* && \text{(Go from 7 to 10)} \\
f_4(8) &= 8^* && \text{(Go from 8 to 10)} \\
f_4(9) &= 9^* && \text{(Go from 9 to 10)}
\end{aligned}
$$

$$
f_3(5) = \min \begin{cases} \max\,[c_{57}, f_4(7)] = 13 & \text{(Go from 5 to 7)} \\ \max\,[c_{58}, f_4(8)] = 8^* & \text{(Go from 5 to 8)} \\ \max\,[c_{59}, f_4(9)] = 10 & \text{(Go from 5 to 9)} \end{cases}
$$

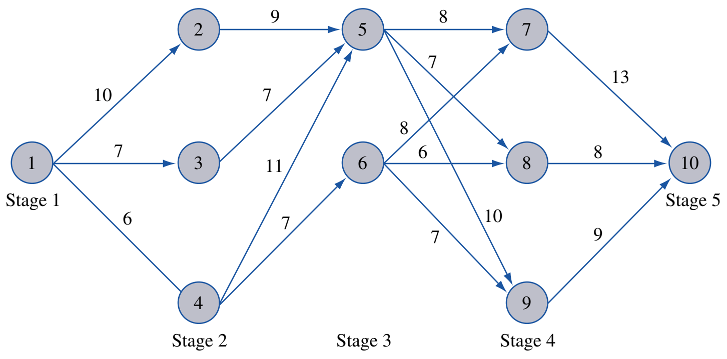


FIGURE 10
Joe's Trip
(Altitudes Given)

$$f_3(6) = \min \begin{cases} \max\,[c_{67}, f_4(7)] = 13 & \text{(Go from 6 to 7)} \\ \max\,[c_{68}, f_4(8)] = 8^* & \text{(Go from 6 to 8)} \\ \max\,[c_{69}, f_4(9)] = 9 & \text{(Go from 6 to 9)} \end{cases}$$

$$f_2(2) = \max\,[c_{25}, f_3(5)] = 9^* \qquad \text{(Go from 2 to 5)}$$

$$f_2(3) = \max\,[c_{35}, f_3(5)] = 8^* \qquad \text{(Go from 3 to 5)}$$

$$f_2(4) = \min \begin{cases} \max\,[c_{45}, f_3(5)] = 11 & \text{(Go from 4 to 5)} \\ \max\,[c_{46}, f_3(6)] = 8^* & \text{(Go from 4 to 6)} \end{cases}$$

$$f_1(1) = \min \begin{cases} \max\,[c_{12}, f_2(2)] = 10 & \text{(Go from 1 to 2)} \\ \max\,[c_{13}, f_2(3)] = 8^* & \text{(Go from 1 to 3)} \\ \max\,[c_{14}, f_2(4)] = 8^* & \text{(Go from 1 to 4)} \end{cases}$$

To determine the optimal strategy, note that Joe can begin by going from city 1 to city 3 or from city 1 to city 4. Suppose Joe begins by traveling to city 3. Then he should choose the arc attaining $f_2(3)$, which means he should next travel to city 5. Then Joe must choose the arc that attains $f_3(5)$, driving next to city 8. Then, of course, he must drive to city 10. Thus, the path 1–3–5–8–10 is optimal, and Joe will encounter a maximum altitude equal to $f_1(1) = 8{,}000$ ft. The reader should verify that the path 1–4–6–8–10 is also optimal.

EXAMPLE 14 **Sales Allocation**

Glueco is planning to introduce a new product in three different regions. Current estimates are that the product will sell well in each region with respective probabilities .6, .5, and .3. The firm has available two top sales representatives that it can send to any of the three regions. The estimated probabilities that the product will sell well in each region when 0, 1, or 2 additional sales reps are sent to a region are given in Table 16. If Glueco wants to maximize the probability that its new product will sell well in all three regions, then where should it assign sales representatives? You may assume that sales in the three regions are independent.

Solution If Glueco had just one region to worry about and wanted to maximize the probability that the new product would sell in that region, then the proper strategy would be clear: Assign both sales reps to the region. We could then work backward and solve a problem in which Glueco's goal is to maximize the probability that the product will sell in two regions. Finally, we could work backward and solve a problem with three regions. We define $f_t(s)$ as the probability that the new product will sell in regions $t, t + 1, \ldots, 3$ if s sales reps are optimally assigned to these regions. Then

$$f_3(2) = .7 \qquad \text{(Assign 2 sales reps to region 3)}$$
$$f_3(1) = .55 \qquad \text{(Assign 1 sales rep to region 3)}$$
$$f_3(0) = .3 \qquad \text{(Assign 0 sales reps to region 3)}$$

TABLE 16
Relation between Regional Sales and Sales Representatives

No. of Additional Sales Representatives	Probability of Selling Well		
	Region 1	Region 2	Region 3
0	.6	.5	.3
1	.8	.7	.55
2	.85	.85	.7

Also, $f_1(2)$ will be the maximum probability that the product will sell well in all three regions. To develop a recursion for $f_2(\cdot)$ and $f_1(\cdot)$, we define p_{tx} to be the probability that the new product sells well in region t if x sales reps are assigned to region t. For example, $p_{21} = .7$. For $t = 1$ and $t = 2$, we then write

$$f_t(s) = \max_x \{p_{tx} f_{t+1}(s - x)\} \tag{21}$$

where x must be a member of $\{0, 1, \ldots, s\}$. To justify (21), observe that if s sales reps are available for regions $t, t + 1, \ldots, 3$ and x sales reps are assigned to region t, then

$$\begin{aligned} p_{tx} &= \text{probability that product sells in region } t \\ f_{t+1}(s - x) &= \text{probability that product sells well in regions } t + 1, \ldots, 3 \end{aligned}$$

Note that the sales in each region are independent. This implies that if x sales reps are assigned to region t, then the probability that the new product sells well in regions t, $t + 1, \ldots, 3$ is $p_{tx} f_{t+1}(s - x)$. We want to maximize this probability, so we obtain (21). Applying (21) yields the following results:

$$f_2(2) = \max \begin{cases} (.5)f_3(2 - 0) = .35 \\ \text{(Assign 0 sales reps to region 2)} \\ (.7)f_3(2 - 1) = .385^* \\ \text{(Assign 1 sales rep to region 2)} \\ (.85)f_3(2 - 2) = .255 \\ \text{(Assign 2 sales reps to region 2)} \end{cases}$$

Thus, $f_2(2) = .385$, and 1 sales rep should be assigned to region 2.

$$f_2(1) = \max \begin{cases} (.5)f_3(1 - 0) = .275^* \\ \text{(Assign 0 sales reps to region 2)} \\ (.7)f_3(1 - 1) = .21 \\ \text{(Assign 1 sales rep to region 2)} \end{cases}$$

Thus, $f_2(1) = .275$, and no sales reps should be assigned to region 2.

$$\begin{aligned} f_2(0) = {} & (.5)f_3(0 - 0) = .15^* \\ & \text{(Assign 0 sales reps to region 2)} \end{aligned}$$

Finally, we are back to the original problem, which is to find $f_1(2)$. Equation (21) yields

$$f_1(2) = \max \begin{cases} (.6)f_2(2 - 0) = .231^* \\ \text{(Assign 0 sales reps to region 1)} \\ (.8)f_2(2 - 1) = .220 \\ \text{(Assign 1 sales rep to region 1)} \\ (.85)f_2(2 - 2) = .1275 \\ \text{(Assign 2 sales reps to region 1)} \end{cases}$$

Thus, $f_1(2) = .231$, and no sales reps should be assigned to region 1. Then Glueco needs to attain $f_2(2 - 0)$, which requires that 1 sales rep be assigned to region 2. Glueco must next attain $f_3(2 - 1)$, which requires that 1 sales rep be assigned to region 3. In summary, Glueco can obtain a .231 probability of the new product selling well in all three regions by assigning 1 sales rep to region 2 and 1 sales rep to region 3.

PROBLEMS

Group A

1 At the beginning of year 1, Sunco Oil owns i_0 barrels of oil reserves. During year $t(t = 1, 2, \ldots, 10)$, the following events occur in the order listed: (1) Sunco extracts and refines x barrels of oil reserves and incurs a cost $c(x)$: (2) Sunco sells year t's extracted and refined oil at a price of p_t dollars per barrel; and (3) exploration for new reserves results in a discovery of b_t barrels of new reserves.

Sunco wants to maximize sales revenues less costs over the next 10 years. Formulate a dynamic programming recursion that will help Sunco accomplish its goal. If Sunco felt that cash flows in later years should be discounted, how should the formulation be modified?

2 At the beginning of year 1, Julie Ripe has D dollars (this includes year 1 income). During each year, Julie earns i dollars and must determine how much money she should consume and how much she should invest in Treasury bills. During a year in which Julie consumes d dollars, she earns a utility of $\ln d$. Each dollar invested in Treasury bills yields $1.10 in cash at the beginning of the next year. Julie's goal is to maximize the total utility she earns during the next 10 years.

- **a** Why might $\ln d$ be a better indicator of Julie's utility than a function such as d^2?
- **b** Formulate a dynamic programming recursion that will enable Julie to maximize the total utility she receives during the next 10 years. Assume that year t revenue is received at the beginning of year t.

3 Assume that during minute t (the current minute is minute 1), the following sequence of events occurs: (1) At the beginning of the minute, x_t customers arrive at the cash register; (2) the store manager decides how many cash registers should be operated during the current minute; (3) if s cash registers are operated and i customers are present (including the current minute's arrivals), $c(s, i)$ customers complete service; and (4) the next minute begins.

A cost of 10¢ is assessed for each minute a customer spends waiting to check out (this time includes checkout time). Assume that it costs $c(s)$ cents to operate s cash registers for 1 minute. Formulate a dynamic programming recursion that minimizes the sum of holding and service costs during the next 60 minutes. Assume that before the first minute's arrivals, no customers are present and that holding cost is assessed at the end of each minute.

4 Develop a dynamic programming formulation of the CSL Computer problem of Section 3.12.

5 To graduate from State University, Angie Warner needs to pass at least one of the three subjects she is taking this semester. She is now enrolled in French, German, and statistics. Angie's busy schedule of extracurricular activities allows her to spend only 4 hours per week on studying. Angie's probability of passing each course depends on the number of hours she spends studying for the course (see Table 17). Use dynamic programming to determine how many hours per week Angie should spend studying each subject. (*Hint:* Explain why maximizing the probability of passing at least one course is equivalent to minimizing the probability of failing all three courses.)

TABLE 17

Hours of Study per Week	Probability of Passing Course		
	French	German	Statistics
0	.20	.25	.10
1	.30	.30	.30
2	.35	.33	.40
3	.38	.35	.44
4	.40	.38	.50

TABLE 18

	No. of Actors Assigned to Component			
Component	0	1	2	3
Warp drive	.30	.55	.65	.95
Solar relay	.40	.50	.70	.90
Candy maker	.45	.55	.80	.98

6 E.T. is about to fly home. For the trip to be successful, the ship's solar relay, warp drive, and candy maker must all function properly. E.T. has found three unemployed actors who are willing to help get the ship ready for takeoff. Table 18 gives, as a function of the number of actors assigned to repair each component, the probability that each component will function properly during the trip home. Use dynamic programming to help E.T. maximize the probability of having a successful trip home.

7 Farmer Jones is trying to raise a prize steer for the Bloomington 4-H show. The steer now weighs w_0 pounds. Each week, farmer Jones must determine how much food to feed the steer. If the steer weighs w pounds at the beginning of a week and is fed p pounds of food during a week, then at the beginning of the next week, the steer will weigh $g(w, p)$ pounds. It costs farmer Jones $c(p)$ dollars to feed the steer p pounds of food during a week. At the end of the 10th week (or equivalently, the beginning of the 11th week), the steer may be sold for $10/lb. Formulate a dynamic programming recursion that can be used to determine how farmer Jones can maximize profit from the steer.

Group B

8 MacBurger has just opened a fast-food restaurant in Bloomington. Currently, i_0 customers frequent MacBurger (we call these loyal customers), and $N - i_0$ customers frequent other fast-food establishments (we call these nonloyal customers). At the beginning of each month, MacBurger must decide how much money to spend on advertising. At the end of a month in which MacBurger spends d dollars on advertising, a fraction $p(d)$ of the loyal customers become nonloyal customers, and a fraction $q(d)$ of the nonloyal customers become loyal customers. During the next 12 months, MacBurger wants to spend D dollars on advertising. Develop a dynamic programming recursion that will enable MacBurger to maximize the number of loyal customers the company will have at the end of month 12. (Ignore the possibility of a fractional number of loyal customers.)

9 Public Service Indiana (PSI) is considering five possible locations to build power plants during the next 20 years. It will cost c_i dollars to build a plant at site i and h_i dollars to operate a site i plant for a year. A plant at site i can supply k_i kilowatt-hours (kwh) of generating capacity. During year t, d_t kwh of generating capacity are required. Suppose that at most one plant can be built during a year, and if it is decided to build a plant at site i during year t, then the site i plant can be used to meet the year t (and later) generating requirements. Initially, PSI has 500,000 kwh of generating capacity available. Formulate a recursion that PSI could use to minimize the sum of building and operating costs during the next 20 years.

10 During month t, a firm faces a demand for d_t units of a product. The firm's production cost during month t consists of two components. First, for each unit produced during month t, the firm incurs a variable production cost of c_t. Second, if the firm's production level during month $t - 1$ is x_{t-1} and the firm's production level during month t is x_t, then during month t, a smoothing cost of $5|x_t - x_{t-1}|$ will be incurred (see Section 16.12 for an explanation of smoothing costs). At the end of each month, a holding cost of h_t per unit is incurred. Formulate a recursion that will enable the firm to meet (on time) its demands over the next 12 months. Assume that at the beginning of the first month, 20 units are in inventory and that last month's production was 20 units. (*Hint:* The state during each month must consist of two quantities.)

11 The state of Transylvania consists of three cities with the following populations: city 1, 1.2 million people; city 2, 1.4 million people; city 3, 400,000 people. The Transylvania House of Representatives consists of three representatives. Given proportional representation, city 1 should have $d_1 = (\frac{1.2}{3}) = 1.2$ representatives; city 2 should have $d_2 = 1.4$ representatives; and city 3 should have $d_3 = 0.40$ representative. Each city must receive an integral number of representatives, so this is impossible. Transylvania has therefore decided to allocate x_i representatives to city i, where the allocation x_1, x_2, x_3 minimizes the maximum discrepancy between the desired and actual number of representatives received by a city. In short, Transylvania must determine x_1, x_2, and x_3 to minimize the largest of the following three numbers: $|x_1 - d_1|$, $|x_2 - d_2|$, $|x_3 - d_3|$. Use dynamic programming to solve Transylvania's problem.

12 A job shop has four jobs that must be processed on a single machine. The due date and processing time for each job are given in Table 19. Use dynamic programming to determine the order in which the jobs should be done so as to minimize the total lateness of the jobs. (The lateness of a job is simply how long after the job's due date the job is completed; for example, if the jobs are processed in the given order, then job 3 will be 2 days late, job 4 will be 4 days late, and jobs 1 and 2 will not be late.)

TABLE 19

Job	Processing Time (Days)	Due Date (Days from Now)
1	2	4
2	4	14
3	6	10
4	8	16

18.7 The Wagner–Whitin Algorithm and the Silver–Meal Heuristic[†]

The inventory example of Section 18.3 is a special case of the *dynamic lot-size model.*

Description of Dynamic Lot-Size Model

1 Demand d_t during period $t (t = 1, 2, \dots, T)$ is known at the beginning of period 1.

2 Demand for period t must be met on time from inventory or from period t production. The cost $c(x)$ of producing x units during any period is given by $c(0) = 0$, and for $x > 0$, $c(x) = K + cx$, where K is a fixed cost for setting up production during a period, and c is the variable per-unit cost of production.

3 At the end of period t, the inventory level i_t is observed, and a holding cost hi_t is incurred. We let i_0 denote the inventory level before period 1 production occurs.

4 The goal is to determine a production level x_i for each period t that minimizes the total cost of meeting (on time) the demands for periods $1, 2, \dots, T$.

[†]This section covers topics that may be omitted with no loss of continuity.

5 There is a limit c_t placed on period t's ending inventory.

6 There is a limit r_t placed on period t's production.

In this section, we consider these first four points. We let x_t = period t production. Period t production can be used to meet period t demand.

EXAMPLE 15 **Dynamic Lot-Size Model**

We now determine an optimal production schedule for a five-period dynamic lot-size model with $K = \$250$, $c = \$2$, $h = \$1$, $d_1 = 220$, $d_2 = 280$, $d_3 = 360$, $d_4 = 140$, and $d_5 = 270$. We assume that the initial inventory level is zero. The solution to this example is given later in this section.

Discussion of the Wagner–Whitin Algorithm

If the dynamic programming approach outlined in Section 18.3 were used to find an optimal production policy for Example 15, we would have to consider the possibility of producing any amount between 0 and $d_1 + d_2 + d_3 + d_4 + d_5 = 1{,}270$ units during period 1. Thus, it would be possible for the period 2 state (period 2's entering inventory) to be $0, 1, \ldots, 1{,}270 - d_1 = 1{,}050$, and we would have to determine $f_2(0), f_2(1), \ldots, f_2(1{,}050)$. Using the dynamic programming approach of Section 18.3 to find an optimal production schedule for Example 15 would therefore require a great deal of computational effort. Fortunately, however, Wagner and Whitin (1958) have developed a method that greatly simplifies the computation of optimal production schedules for dynamic lot-size models. Lemmas 1 and 2 are necessary for the development of the Wagner–Whitin algorithm.

LEMMA 1

Suppose it is optimal to produce a positive quantity during a period t. Then for some $j = 0, 1, \ldots, T - t$, the amount produced during period t must be such that after period t's production, a quantity $d_t + d_{t+1} + \cdots + d_{t+j}$ will be in stock. In other words, if production occurs during period t, we must (for some j) produce an amount that exactly suffices to meet the demands for periods $t, t + 1, \ldots, t + j$.

Proof If the lemma is false, then for some t, some $j = 0, 1, \ldots, T - t - 1$, and some x satisfying $0 < x < d_{t+j+1}$, period t production must bring the stock level to $d_t + d_{t+1} + \cdots + d_{t+j} + x$, and at the beginning of period $t + j + 1$, our inventory level would be $x < d_{t+j+1}$. Thus, production must occur during period $t + j + 1$. By deferring production of x units from period t to period $t + j + 1$ (with all other production levels unchanged), we save $h(j + 1)x$ in holding costs while incurring no additional setup costs (because production is already occurring during period $t + j + 1$). Thus, it cannot have been optimal to bring our period t stock level to $d_t + d_{t+1} + \cdots + d_{t+j} + x$. This contradiction proves the lemma.

LEMMA 2

If it is optimal to produce anything during period t, then $i_{t-1} < d_t$. In other words, production cannot occur during period t unless there is insufficient stock to meet period t demand.

Proof If the lemma is false, there must be an optimal policy that (for some t) has $x_t > 0$ and $i_{t-1} \geq d_t$. If this is the case, then by deferring the period t production of x_t units to period $t + 1$, we save hx_t in holding costs and possibly K (if the optimal policy produces during period $t + 1$) in setup costs. Thus, any production schedule having $x_t > 0$ and $i_{t-1} \geq d_t$ cannot be optimal.

Lemma 2 shows that no production will occur until the first period t for which $i_{t-1} < d_t$, so production must occur during period t (or else period t's demand would not be met on time). Lemma 1 now implies that for some $j = 0, 1, \ldots, T - t$, period t production will be such that after period t's production, on-hand stock will equal $d_t + d_{t+1} + \cdots + d_{t+j}$. Then Lemma 2 implies that no production can occur until period $t + j + 1$. Since the entering inventory level for period $t + j + 1$ will equal zero, production must occur during period $t + j + 1$. During period $t + j + 1$, Lemma 1 implies that period $t + j + 1$ production will (for some k) equal $d_{t+j+1} + d_{t+j+2} + \cdots + d_{t+j+k}$ units. Then period $t + j + k + 1$ will begin with zero inventory, and production again occurs, and so on. *With the possible exception of the first period, production will occur only during periods in which beginning inventory is zero, and during each period in which beginning inventory is zero (and $d_t \neq 0$), production must occur.*

Using this insight, Wagner and Whitin developed a recursion that can be used to determine an optimal production policy. We assume that the initial inventory level is zero. (See Problem 1 at the end of this section if this is not the case.) Define f_t as the minimum cost incurred during periods $t, t + 1, \ldots, T$, given that at the beginning of period t, the inventory level is zero. Then $f_1, f_2, \ldots, f_T$ must satisfy

$$f_t = \min_{j=0,\,1,\,2,\ldots,\,T-t} (c_{tj} + f_{t+j+1}) \tag{22}$$

where $f_{T+1} = 0$ and c_{tj} is the total cost incurred during periods $t, t + 1, \ldots, t + j$ if production during period t is exactly sufficient to meet demands for periods $t, t + 1, \ldots, t + j$. Thus,

$$c_{tj} = K + c(d_t + d_{t+1} + \cdots + d_{t+j}) + h[jd_{t+j} + (j - 1)d_{t+j-1} + \cdots + d_{t+1}]$$

where K is the setup cost incurred during period t, $c(d_t + d_{t+1} + \cdots + d_{t+j})$ is the variable production cost incurred during period t, and $h[jd_{t+j} + (j - 1)d_{t+j-1} + \cdots + d_{t+1}]$ is the holding cost incurred during periods $t, t + 1, \ldots, t + j$. For example, an amount d_{t+j} of period t production will be held in inventory for j periods (during periods $t, t + 1, \ldots, t + j - 1$), thereby incurring a holding cost of hjd_{t+j}.

To find an optimal production schedule by the Wagner–Whitin algorithm, begin by using (22) to find f_T. Then use (22) to compute $f_{T-1}, f_{T-2}, \ldots, f_1$. Once f_1 has been determined, an optimal production schedule may be easily obtained.

EXAMPLE 15 Dynamic Lot-Size Model (continued)

Solution To illustrate the Wagner–Whitin algorithm, we find an optimal production schedule for Example 15. The computations follow.

$$f_6 = 0$$

$$f_5 = 250 + 2(270) + f_6 = 790^* \qquad \text{(Produce for period 5)}$$

If we begin period 5 with zero inventory, we should produce enough during period 5 to meet period 5 demand.

$$f_4 = \min \begin{cases} 250 + 2(140) + f_5 = 1{,}320^* \\ \text{(Produce for period 4)} \\ 250 + 2(140 + 270) + 270 + f_6 = 1{,}340 \\ \text{(Produce for periods 4, 5)} \end{cases}$$

If we begin period 4 with zero inventory, we should produce enough during period 4 to meet the demand for period 4.

$$f_3 = \min \begin{cases} 250 + 2(360) + f_4 = 2{,}290 \\ \text{(Produce for period 3)} \\ 250 + 2(360 + 140) + 140 + f_5 = 2{,}180^* \\ \text{(Produce for periods 3, 4)} \\ 250 + 2(360 + 140 + 270) + 140 + 2(270) + f_6 = 2{,}470 \\ \text{(Produce for periods 3, 4, 5)} \end{cases}$$

If we begin period 3 with zero inventory, we should produce enough during period 3 to meet the demand for periods 3 and 4.

$$f_2 = \min \begin{cases} 250 + 2(280) + f_3 = 2{,}990^* \\ \text{(Produce for period 2)} \\ 250 + 2(280 + 360) + 360 + f_4 = 3{,}210 \\ \text{(Produce for periods 2, 3)} \\ 250 + 2(280 + 360 + 140) + 360 + 2(140) + f_5 = 3{,}240 \\ \text{(Produce for periods 2, 3, 4)} \\ 250 + 2(280 + 360 + 140 + 270) + 360 + 2(140) + 3(270) + f_6 = 3{,}800 \\ \text{(Produce for periods 2, 3, 4, 5)} \end{cases}$$

If we begin period 2 with zero inventory, we should produce enough during period 2 to meet the demand for period 2.

$$f_1 = \min \begin{cases} 250 + 2(220) + f_2 = 3{,}680^* \\ \text{(Produce for period 1)} \\ 250 + 2(220 + 280) + 280 + f_3 = 3{,}710 \\ \text{(Produce for periods 1, 2)} \\ 250 + 2(220 + 280 + 360) + 280 + 2(360) + f_4 = 4{,}290 \\ \text{(Produce for periods 1, 2, 3)} \\ 250 + 2(220 + 280 + 360 + 140) + 280 + 2(360) + 3(140) + f_5 = 4{,}460 \\ \text{(Produce for periods 1, 2, 3, 4)} \\ 250 + 2(220 + 280 + 360 + 140 + 270) + 280 \\ \quad + 2(360) + 3(140) + 4(270) + f_6 = 5{,}290 \\ \text{(Produce for periods 1, 2, 3, 4, 5)} \end{cases}$$

If we begin period 1 with zero inventory, it is optimal to produce $d_1 = 220$ units during period 1; then we begin period 2 with zero inventory. Since f_2 is attained by producing period 2's demand, we should produce $d_2 = 280$ units during period 2; then we enter period 3 with zero inventory. Since f_3 is attained by meeting the demands for periods 3 and 4, we produce $d_3 + d_4 = 500$ units during period 3; then we enter period 5 with zero inventory and produce $d_5 = 270$ units during period 5. The optimal production schedule will incur at total cost of $f_1 = \$3{,}680$.

For Example 15, any optimal production schedule must produce exactly $d_1 + d_2 + d_3 + d_4 + d_5 = 1{,}270$ units, incurring variable production costs of $2(1{,}270) = \$2{,}540$. Thus, in computing the optimal production schedule, we may always ignore the variable production costs. This substantially simplifies the calculations.

The Silver–Meal Heuristic

The Silver–Meal (S–M) heuristic involves less work than the Wagner–Whitin algorithm and can be used to find a near-optimal production schedule. The S–M heuristic is based on the fact that our goal is to minimize average cost per period (for the reasons stated, variable production costs may be ignored). Suppose we are at the beginning of period 1 and are trying to determine how many periods of demand should be satisfied by period 1's production. During period 1, if we produce an amount sufficient to meet demand for the next t periods, then a cost of $TC(t) = K + HC(t)$ will be incurred (ignoring variable production costs). Here, $HC(t)$ is the holding cost incurred during the next t periods (including the current period) if production during the current period is sufficient to meet demand for the next t periods.

Let $AC(t) = \frac{TC(t)}{t}$ be the average per-period cost incurred during the next t periods. Since $\frac{1}{t}$ is a decreasing convex function of t, as t increases, $\frac{K}{t}$ decreases at a decreasing rate. In most cases, $\frac{HC(t)}{t}$ tends to be an increasing function of t (see Problem 4 at the end of this section). Thus, in most situations, an integer t^* can be found such that for $t < t^*$, $AC(t + 1) \le AC(t)$ and $AC(t^* + 1) \ge AC(t^*)$. The S–M heuristic recommends that period 1's production be sufficient to meet the demands for periods $1, 2, \ldots, t^*$ (if no t^* exists, period 1 production should satisfy the demand for periods $1, 2, \ldots, T$). Since t^* is a local (and perhaps a global) minimum for $AC(t)$, it seems reasonable that producing $d_1 + d_2 + \cdots + d_{t^*}$ units during period 1 will come close to minimizing the average per-period cost incurred during periods $1, 2, \ldots, t^*$. Next we apply the S–M heuristic while considering period $t^* + 1$ as the initial period. We find that during period $t^* + 1$, the demand for the next t_1^* periods should be produced. Continue in this fashion until the demand for period T has been produced.

To illustrate, we apply the S–M heuristic to Example 15. We have

$$TC(1) = 250 \qquad AC(1) = \frac{250}{1} = 250$$

$$TC(2) = 250 + 280 = 530 \qquad AC(2) = \frac{530}{2} = 265$$

Since $AC(2) \ge AC(1)$, $t^* = 1$, and the S–M heuristic dictates that we produce $d_1 = 220$ units during period 1. Then

$$TC(1) = 250 \qquad AC(1) = \frac{250}{1} = 250$$

$$TC(2) = 250 + 360 = 610 \qquad AC(2) = \frac{610}{2} = 305$$

Since $AC(2) \ge AC(1)$, the S–M heuristic recommends producing $d_2 = 280$ units during period 2. Then

$$TC(1) = 250 \qquad AC(1) = \frac{250}{1} = 250$$

$$TC(2) = 250 + 140 = 390 \qquad AC(2) = \frac{390}{2} = 195$$

$$TC(3) = 250 + 2(270) + 140 = 930 \qquad AC(3) = \frac{930}{3} = 310$$

Since $AC(3) \geq AC(2)$, period 3 production should meet the demand for the next two periods (periods 3 and 4). During period 3, we should produce $d_3 + d_4 = 500$ units. This brings us to period 5. Period 5 is the final period, so $d_5 = 270$ units should be produced during period 5.

For Example 15 (and many other dynamic lot-size problems), the S–M heuristic yields an optimal production schedule. In extensive testing, the S–M heuristic usually yielded a production schedule costing less than 1% above the optimal policy obtained by the Wagner–Whitin algorithm (see Peterson and Silver (1998)).

PROBLEMS

Group A

1 For Example 15, suppose we had an inventory of 200 units. What would be the optimal production schedule? What if the initial inventory were 400 units?

2 Use the Wagner–Whitin and Silver–Meal methods to find production schedules for the following dynamic lot-size problem: $K = \$50$, $h = \$0.40$, $d_1 = 10$, $d_2 = 60$, $d_3 = 20$, $d_4 = 140$, $d_5 = 90$.

3 Use the Wagner–Whitin and Silver–Meal methods to find production schedules for the following dynamic lot-size problem: $K = \$30$, $h = \$1$, $d_1 = 40$, $d_2 = 60$, $d_3 = 10$, $d_4 = 70$, $d_5 = 20$.

Group B

4 Explain why $HC(t)/t$ tends to be an increasing function of t.

18.8 Using Excel to Solve Dynamic Programming Problems[†]

In earlier chapters, we have seen that any LP problem can be solved with LINDO or LINGO, and any NLP can be solved with LINGO. Unfortunately, no similarly user-friendly package can be used to solve dynamic programming problems. LINGO can be used to solve DP problems, but student LINGO can only handle a very small problem. Fortunately, Excel can often be used to solve DP problems. Our three illustrations solve a knapsack problem (Example 6), a resource-allocation problem (Example 5), and an inventory problem (Example 4).

Solving Knapsack Problems on a Spreadsheet

Recall the knapsack problem of Example 6. The question is how to (using three types of items) fill a 10-lb knapsack and obtain the maximum possible benefit. Recall that $g(w)$ = maximum benefit that can be obtained from a w-lb knapsack. Recall that

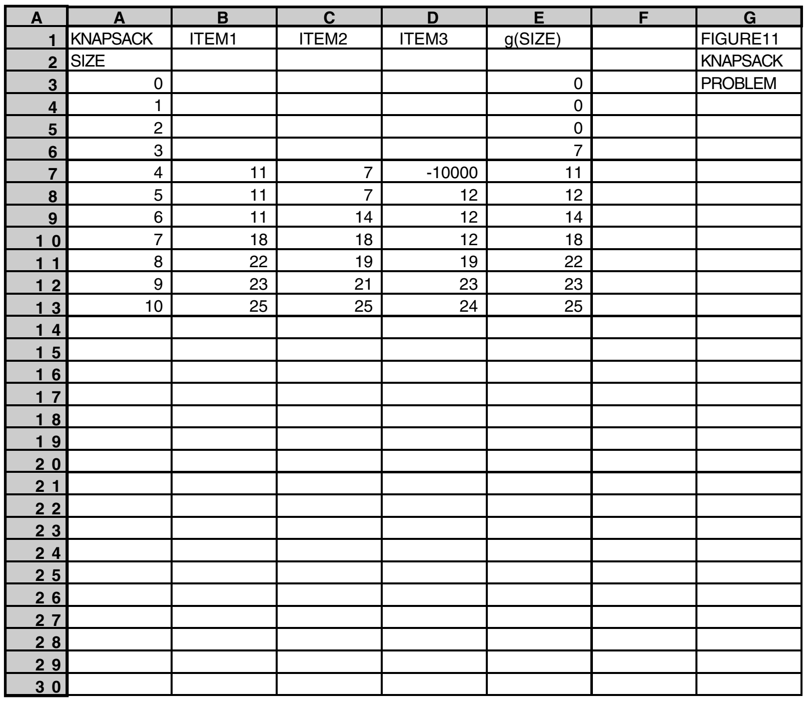
$$g(w) = \max_j \{b_j + g(w - w_j)\} \tag{8}$$

where b_j = benefit from a type j item and w_j = weight of a type j item.

Dpknap.xls

In each row of the spreadsheet (see Figure 11 or file Dpknap.xls) we compute $g(w)$ for various values of w. We begin by entering $g(0) = g(1) = g(2) = 0$ and $g(3) = 7$; [$g(3) = 7$ follows because a 3-lb item is the only item that will fit in a 3-lb knapsack]. The

[†]This section covers topics that may be omitted with no loss of continuity.



A	A	B	C	D	E	F	G
1	KNAPSACK	ITEM1	ITEM2	ITEM3	g(SIZE)		FIGURE11
2	SIZE						KNAPSACK
3	0				0		PROBLEM
4	1				0		
5	2				0		
6	3				7		
7	4	11	7	-10000	11		
8	5	11	7	12	12		
9	6	11	14	12	14		
10	7	18	18	12	18		
11	8	22	19	19	22		
12	9	23	21	23	23		
13	10	25	25	24	25		
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							

FIGURE 11
Knapsack Problem

columns labeled ITEM1, ITEM2, and ITEM3 correspond to the terms $j = 1, 2, 3$, respectively, in (8). Thus, in the ITEM1 column, we should enter a formula to compute $b_1 + g(w - w_1)$; in the ITEM2 column, we should enter a formula to compute $b_2 + g(w - w_2)$; in the ITEM3 column, we should enter a formula to compute $b_3 + g(w - w_3)$. The only exception to this occurs when a w_j-lb item will not fit in a w-lb knapsack. In this situation, we enter a very negative number (such as 10,000) to ensure that a w_j-lb item will not be considered.

More specifically, in row 7, we want to compute $g(4)$. To do this, we enter the following formulas:

B7: 11 + E3 [This is $b_1 + g(4 - w_1)$]

C7: 7 + E4 [This is $b_2 + g(4 - w_2)$]

D7: −10,000 (This is because a 5-lb item will not fit in a 4-lb knapsack)

In E7, we compute $g(4)$ by entering the formula =MAX(B7:D7). In row 8, we compute $g(5)$ by entering the following formulas:

B8: 11 + E4

C8: 7 + E5

D8: 12 + E3

To compute $g(5)$, we enter =MAX(B8:D8) in E8. Now comes the fun part! Simply copy the formulas from the range B8:E8 to B8:E13. Then $g(10)$ will be computed in E13. We see that $g(10) = 25$. Because both item 1 and item 2 attain $g(10)$, we may begin filling a knapsack with a Type 1 or Type 2 item. We choose to begin with a Type 1 item. This leaves us with $10 - 4 = 6$ lb to fill. From row 9 we find that $g(6) = 14$ is attained by a Type 2 item. This leaves us with $6 - 3 = 3$ lb to fill. We also use a Type 2 item to attain $g(3) = 7$. This leaves us with 0 lb. Thus, we conclude that we can obtain 25 units of benefit by filling a 10-lb knapsack with two Type 2 items and one Type 1 item.

By the way, if we had been interested in filling a 100-lb knapsack, we would have copied the formulas from B8:E8 to B8:E103.

Solving a General Resource-Allocation Problem on a Spreadsheet

Solving a nonknapsack resource-allocation problem on a spreadsheet is more difficult. To illustrate, consider Example 5 in which we have \$6,000 to allocate between three investments. Define $f_t(d)$ = maximum NPV obtained from investments $t, \ldots, 3$ given that d (in thousands) dollars are available for investments $t, \ldots, 3$. Then we may write

$$f_t(d) = \max_{0 \le x \le d} \{r_t(x) + f_{t+1}(d - x)\} \tag{10}$$

where $f_4(d) = 0(d = 0, 1, 2, 3, 4, 5, 6)$, $r_t(x)$ = NPV obtained if x (in thousands) dollars are invested in investment t, and the maximization in (10) is only taken over integral values for d. Our subsequent discussion will be simplified if we define $J_t(d, x) = r_t(x) + f_{t+1}(d - x)$ and rewrite (10) as

$$f_t(d) = \max_{0 \le x \le d} \{J_t(d, x)\} \tag{10′}$$

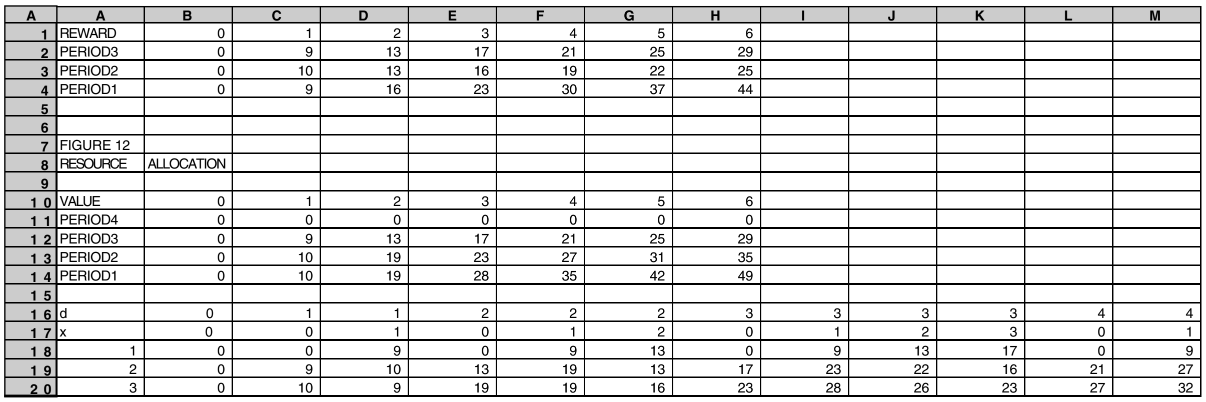
Dpresour.xls

We begin the construction of the spreadsheet (Figure 12 and file Dpresour.xls) by entering the $r_t(x)$ in A1:H4. For example, $r_2(3) = 16$ is entered in E3. In rows 18–20, we have set up the computations to compute the $J_t(d, x)$. These computations require using the Excel =**HLOOKUP** command to look up the values of $r_t(x)$ (in rows 2–4) and $f_{t+1}(d - x)$ (in rows 11–14). For example, to compute $J_3(3, 1)$, we enter the following formula in I18:

=HLOOKUP(I\$17,\$B\$1:\$H\$4,\$A18+1)
+ HLOOKUP(I\$16-I\$17,\$B\$10:\$H\$14,\$A18+1)

The portion =HLOOKUP(I\$17,\$B\$1:\$H\$4,\$A18+1) of the formula in cell I18 finds the column in B1:H4 whose first entry matches I17. Then we pick off the entry in row A18 + 1 of that column. This returns $r_3(1) = 9$. Note that H stands for horizontal lookup. The portion HLOOKUP(I\$16-I\$17,\$b\$10:\$h\$14,\$A18+1) finds the column in B10:H14 whose first entry matches I16-I17. Then we pick off the entry in row A18 + 1 of that column. This yields $f_4(3 - 1) = 0$.

FIGURE 12
Resource Allocation



A	A	B	C	D	E	F	G	H	I	J	K	L	M
1	REWARD	0	1	2	3	4	5	6					
2	PERIOD3	0	9	13	17	21	25	29					
3	PERIOD2	0	10	13	16	19	22	25					
4	PERIOD1	0	9	16	23	30	37	44					
5													
6													
7	FIGURE 12												
8	RESOURCE	ALLOCATION											
9													
10	VALUE	0	1	2	3	4	5	6					
11	PERIOD4	0	0	0	0	0	0	0					
12	PERIOD3	0	9	13	17	21	25	29					
13	PERIOD2	0	10	19	23	27	31	35					
14	PERIOD1	0	10	19	28	35	42	49					
15													
16	d	0	1	1	2	2	2	3	3	3	3	4	4
17	x	0	0	1	0	1	2	0	1	2	3	0	1
18	1	0	0	9	0	9	13	0	9	13	17	0	9
19	2	0	9	10	13	19	13	17	23	22	16	21	27
20	3	0	10	9	19	19	16	23	28	26	23	27	32

FIGURE 12
(Continued)

A	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1													
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16	4	4	4	5	5	5	5	5	5	6	6	6	6
17	2	3	4	0	1	2	3	4	5	0	1	2	3
18	13	17	21	0	9	13	17	21	25	0	9	13	17
19	26	25	19	25	31	30	29	28	22	29	35	34	33
20	35	33	30	31	36	39	42	40	37	35	40	43	46

A	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16	6	6	6	0	1	2	3	4	5	6	
17	4	5	6	ft(0)	ft(1)	ft(2)	ft(3)	ft(4)	ft(5)	ft(6)	t
18	21	25	29	0	9	13	17	21	25	29	3
19	32	31	25	0	10	19	23	27	31	35	2
20	49	47	44	0	10	19	28	35	42	49	1

We now copy any of the $J_t(d, x)$ formulas (such as the one in I18) to the range B18:AC20.

The $f_t(d)$ are computed in AD18 : AJ20. We begin by manually entering in AD18:AJ18 the formulas used to compute $f_3(0), f_3(1), \ldots, f_3(6)$. These formulas are as follows:

AD18:	0	(Computes $f_3(0)$)
AE18:	=MAX(C18:D18)	(Computes $f_3(1)$)
AF18:	=MAX(E18:G18)	(Computes $f_3(2)$)
AG18:	=MAX(H18:K18)	(Computes $f_3(3)$)
AH18:	=MAX(L18:P18)	(Computes $f_3(4)$)
AI18:	=MAX(Q18:V18)	(Computes $f_3(5)$)
AJ18:	=MAX(W18:AC18)	(Computes $f_3(6)$)

We now copy these formulas from the range AD18:AJ18 to the range AD18:AJ20.

For our spreadsheet to work we must be able to compute the $J_t(d, x)$ by looking up the appropriate value of $f_t(d)$ in rows 11–14. Thus, in B11:H11, we enter a zero in each cell [because $f_4(d) = 0$ for all d]. In B12, we enter =AD18 [this is the cell in which $f_3(0)$ is computed]. We now copy this formula to the range B12 : H14.

Note that rows 11–14 of our spreadsheet are defined in terms of rows 18–20, and rows 18–20 are defined in terms of rows 11–14. This creates **circularity** or **circular references** in our spreadsheet. To resolve the circular references in this (or any) spreadsheet, simply select Tools, Options, Calculations and select the Iteration box. This will cause Excel to resolve all circular references until the circularity is resolved.

To determine how \$6,000 should be allocated to the three investments, note that $f_1(6) = 49$. Because $f_1(6) = J_1(6, 4)$, we allocate \$4,000 to investment 1. Then we must find $f_2(6 - 4) = 19 = J_2(2, 1)$. We allocate \$1,000 to investment 2. Finally, we find that $f_3(2 - 1) = J_3(1, 1)$ and allocate \$1,000 to investment 3.

Solving an Inventory Problem on a Spreadsheet

We now show how to determine an optimal production policy for Example 4. An important aspect of this production problem is that each month's ending inventory must be between 0 and 4 units. We can ensure that this occurs by manually determining the allowable actions in each state. We will design our spreadsheet to ensure that the ending inventory for each month must be between 0 and 4 inclusive.

Dpinv.xls

Our first step in setting up the spreadsheet (Figure 13, file Dpinv.xls) is to enter the production cost for each possible production level (0, 1, 2, 3, 4, 5) in B1:G2. Then we define $f_t(i)$ to be the minimum cost incurred in meeting demands for months $t, t + 1, \ldots, 4$ when i units are on hand at the beginning of month t. If d_t is month t's demand, then for $t = 1, 2, 3, 4$ we may write

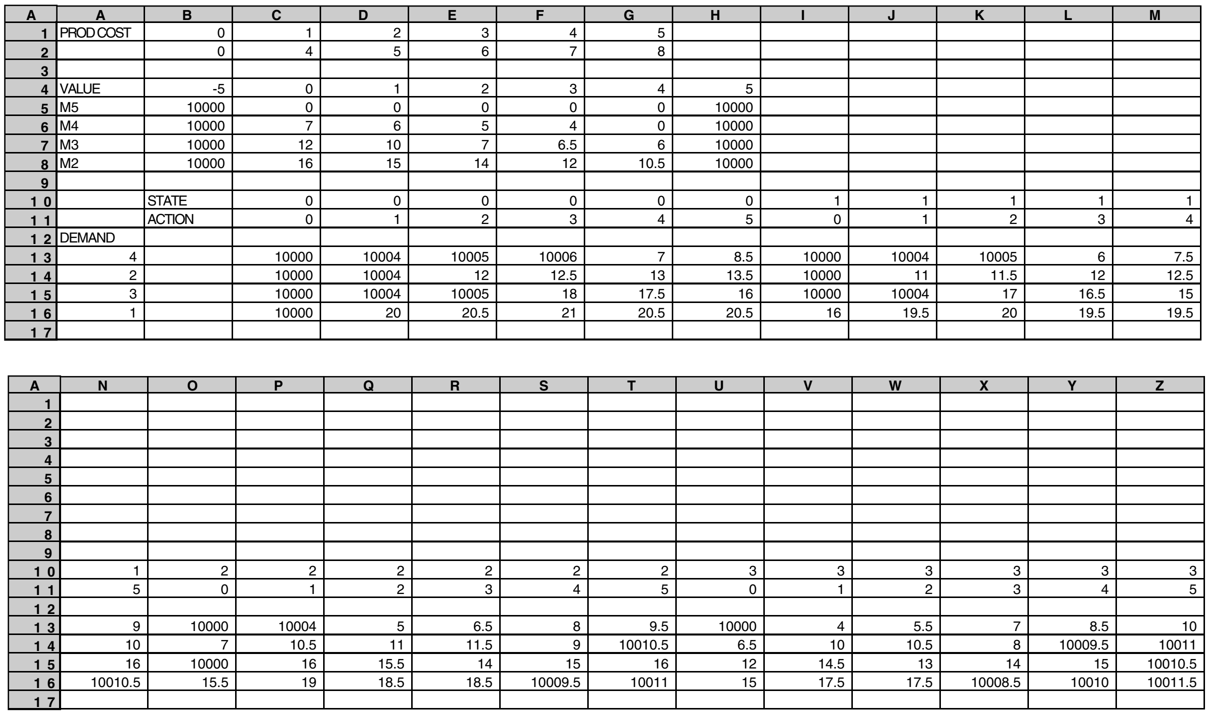
$$f_t(i) = \min_{x|0 \leq i+x-d_t \leq 4} \{.5(i + x - d_t) + c(x) + f_{t+1}(i + x - d_t)\} \tag{23}$$

where $c(x)$ = cost of producing x units during a month, and $f_5(i) = 0$ for ($i = 0, 1, 2, 3, 4$).

If we define $J_t(i, x) = .5(i + x - d_t) + c(x) + f_{t+1}(i + x - d_t)$ we may write

$$f_t(i) = \min_{x|0 \leq i+x-d_t \leq 4} \{J_t(i, x)\}$$

FIGURE 13
Inventory Example



A	A	B	C	D	E	F	G	H	I	J	K	L	M
1	PROD COST	0	1	2	3	4	5						
2		0	4	5	6	7	8						
3													
4	VALUE	-5	0	1	2	3	4	5					
5	M5	10000	0	0	0	0	0	10000					
6	M4	10000	7	6	5	4	0	10000					
7	M3	10000	12	10	7	6.5	6	10000					
8	M2	10000	16	15	14	12	10.5	10000					
9													
10		STATE	0	0	0	0	0	0	1	1	1	1	1
11		ACTION	0	1	2	3	4	5	0	1	2	3	4
12	DEMAND												
13	4		10000	10004	10005	10006	7	8.5	10000	10004	10005	6	7.5
14	2		10000	10004	12	12.5	13	13.5	10000	11	11.5	12	12.5
15	3		10000	10004	10005	18	17.5	16	10000	10004	17	16.5	15
16	1		10000	20	20.5	21	20.5	20.5	16	19.5	20	19.5	19.5
17													

A	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1													
2													
3													
4													
5													
6													
7													
8													
9													
10	1	2	2	2	2	2	2	3	3	3	3	3	3
11	5	0	1	2	3	4	5	0	1	2	3	4	5
12													
13	9	10000	10004	5	6.5	8	9.5	10000	4	5.5	7	8.5	10
14	10	7	10.5	11	11.5	9	10010.5	6.5	10	10.5	8	10009.5	10011
15	16	10000	16	15.5	14	15	16	12	14.5	13	14	15	10010.5
16	10010.5	15.5	19	18.5	18.5	10009.5	10011	15	17.5	17.5	10008.5	10010	10011.5
17													

FIGURE 13
(Continued)

A	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1												
2												
3												
4												
5												
6												
7												
8												
9												
10	4	4	4	4	4	4						
11	0	1	2	3	4	5	F(0)	F(1)	F(2)	F(3)	F(4)	
12												
13	0	4.5	6	7.5	9	10010.5	7	6	5	4	0	1
14	6	9.5	7	10008.5	10010	10011.5	12	10	7	6.5	6	2
15	10.5	12	13	14	10009.5	10011	16	15	14	12	10.5	3
16	13.5	16.5	10007.5	10009	10010.5	10012	20	16	15.5	15	13.5	4
17												

Next we compute $J_t(i, x)$ in A13:AF16. For example, to compute $J_4(0, 2)$, we enter the following formula in E13:

=HLOOKUP(E\$11,\$B\$1:\$G\$2,2)
+.5*+1MAX(E\$10+E\$11−\$A13,0)
+HLOOKUP(E\$10+E\$11−\$A13,\$B\$4:\$H\$8,1+\$AL13)

The first term in this sum yields $c(x)$ (this is because E\$11 is the production level). The second term gives the holding cost for the month (this is because E\$10+E\$11−\$A13 gives the month's ending inventory). The final term yields $f_{t+1}(i + x - d_t)$. This is because E\$10+E\$11−\$A13 is the beginning inventory for month $t + 1$. The reference to 1+\$AL13 in the final term ensures that we look up the value of $f_{t+1}(i + x - d_t)$ in the correct row [the values of the $f_{t+1}(\ )$ will be tabulated in C5:G8]. Copying the formula in E13 to the range C13:AF16 computes all the $J_t(i, x)$.

In AG13:AK16, we compute the $f_t(d)$. To begin, we enter the following formulas in cells AG13:AK13:

AG13:	=MIN(C13:H13)	[Computes $f_4(0)$]
AH13:	=MIN(I13:N13)	[Computes $f_4(1)$]
AI13:	=MIN(O13:T13)	[Computes $f_4(2)$]
AJ13:	=MIN(U13:Z13)	[Computes $f_4(3)$]
AK13:	=MIN(AA13:AF13)	[Computes $f_4(4)$]

To compute all the $f_t(i)$, we now copy from the range AG13:AK13 to the range AG13:AK16. For this to be successful, we need to have the correct values of the $f_t(i)$ in B5:H8. In columns B and H of rows 5–8, we enter 10,000 (or any large positive number). This ensures that it is very costly to end a month with an inventory that is negative or that exceeds 4. This will ensure that each month's ending inventory is between 0 and 4 inclusive. In the range C5:G5, we enter a 0 in each cell. This is because $f_5(i) = 0$ for $i = 0$, 1, 2, 3, 4. In cell C6, we enter +AG13; this enters the value of $f_1(0)$. By copying this formula to the range C6:G8, we have created a table of the $f_t(d)$, which can be used (in rows 13–16) to look up the $f_t(d)$.

As with the spreadsheet we used to solve Example 5, our current spreadsheet exhibits circular references. This is because rows 6–8 refer to rows 13–16, and rows 13–16 refer to rows 6–8. Pressing F9 several times, however, resolves the circular references. You also can resolve circular references by selecting Tools, Options, Calculations and checking the Iterations box.

For any initial inventory level, we can now compute the optimal production schedule. For example, suppose the inventory at the beginning of month 1 is 0. Then $f_1(0) = 20 = J_1(0, 1)$. Thus, it is optimal to produce 1 unit during month 1. Now we seek $f_2(0 + 1 - 1) = 16 = J_2(0, 5)$, so we produce 5 units during month 2. Then we seek $f_3(0 + 5 - 3) = 7 = J_3(2, 0)$, so we produce 0 units during month 3. Solving $f_4(2 + 0 - 2) = J_4(0, 4)$, we produce 4 units during month 4.

PROBLEMS

Group A

1 Use a spreadsheet to solve Problem 2 of Section 18.3.

2 Use a spreadsheet to solve Problem 4 of Section 18.4.

3 Use a spreadsheet to solve Problem 5 of Section 18.4.

SUMMARY

Dynamic programming solves a relatively complex problem by decomposing the problem into a series of simpler problems. First we solve a one-stage problem, then a two-stage problem, and finally a T-stage problem (T = total number of stages in the original problem).

In most applications, a decision is made at each stage (t = current stage), a reward is earned (or a cost is incurred) at each stage, and we go on to the stage $t + 1$ state.

Working Backward

In formulating dynamic programming recursions by working backward, it is helpful to remember that in most cases:

1 The **stage** is the mechanism by which we build up the problem.

2 The **state** at any stage gives the information needed to make the correct decision at the current stage.

3 In most cases, we must determine how the reward received (or cost incurred) during the current stage depends on the stage t decision, the stage t state, and the value of t.

4 We must also determine how the stage $t + 1$ state depends on the stage t decision, the stage t state, and the value of t.

5 If we define (for a minimization problem) $f_t(i)$ as the minimum cost incurred during stages $t, t + 1, \ldots, T$, given that the stage t state is i, then (in many cases) we may write $f_t(i) = \min \{(\text{cost during stage } t) + f_{t+1}(\text{new state at stage } t + 1)\}$, where the minimum is over all decisions allowable in state i during stage t.

6 We begin by determining all the $f_T(\cdot)$'s, then all the $f_{T-1}(\cdot)$'s, and finally f_1 (the initial state).

7 We then determine the optimal stage 1 decision. This leads us to a stage 2 state, at which we determine the optimal stage 2 decision. We continue in this fashion until the optimal stage T decision is found.

Wagner–Whitin Algorithm and Silver–Meal Heuristic for Dynamic Lot-Size Model

A periodic review inventory model in which each period's demand is known at the beginning of the problem is a **dynamic lot-size model.** A cost-minimizing production or ordering policy may be found via a backward recursion, a forward recursion, the Wagner–Whitin algorithm, or the Silver–Meal heuristic.

The Wagner–Whitin algorithm uses the fact that production occurs during a period if and only if the period's beginning inventory is zero. The decision during such a period is the number of consecutive periods of demand that production should meet.

During a period in which beginning inventory is zero, the Silver–Meal heuristic computes the average cost per period (setup plus holding) incurred in meeting the demand during the next k periods. If k^* minimizes this average cost, then the next k^* periods of demand should be met by the current period's production.

Computational Considerations

Dynamic programming is much more efficient than explicit enumeration of the total cost associated with each possible set of decisions that may be chosen during the T stages. Unfortunately, however, many practical applications of dynamic programming involve very large state spaces, and in these situations, considerable computational effort is required to determine optimal decisions.

REVIEW PROBLEMS

Group A

1 In the network in Figure 14, find the shortest path from node 1 to node 10 and the shortest path from node 2 to node 10.

2 A company must meet the following demands on time: month 1, 1 unit; month 2, 1 unit; month 3, 2 units; month 4, 2 units. It costs $4 to place an order, and a $2 per-unit holding cost is assessed against each month's ending inventory. At the beginning of month 1, 1 unit is available. Orders are delivered instantaneously.

a Use a backward recursion to determine an optimal ordering policy.

b Use the Wagner–Whitin method to determine an optimal ordering policy.

c Use the Silver–Meal heuristic to determine an ordering policy.

3 Reconsider Problem 2, but now suppose that demands need not be met on time. Assume that all lost demand is backlogged and that a $1 per-unit shortage cost is assessed against the number of shortages incurred during each month. All demand must be met by the end of month 4. Use dynamic programming to determine an ordering policy that minimizes total cost.

4 Indianapolis Airlines has been told that it may schedule six flights per day departing from Indianapolis. The destination of each flight may be New York, Los Angeles, or Miami. Table 20 shows the contribution to the company's profit from any given number of daily flights from Indianapolis to each possible destination. Find the optimal number of flights that should depart Indianapolis for each destination. How would the answer change if the airline were restricted to only four daily flights?

FIGURE 14

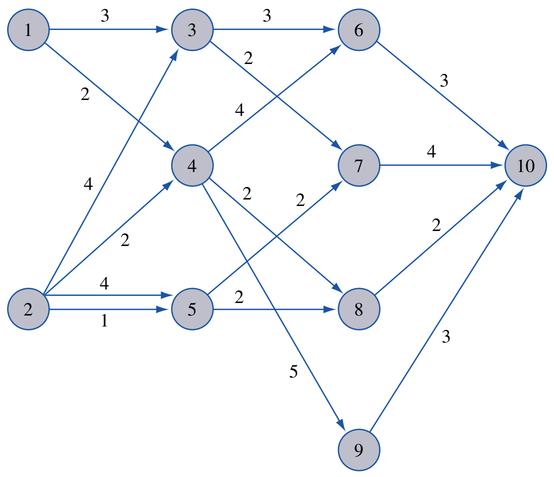


TABLE 20

Destination	Profit per Flight ($) Number of Planes 1	2	3	4	5	6
New York	80	150	210	250	270	280
Los Angeles	100	195	275	325	300	250
Miami	90	180	265	310	350	320

5 I am working as a cashier at the local convenience store. A customer's bill is $1.09, and he gives me $2.00. I want to give him change using the smallest possible number of coins. Use dynamic programming to determine how to give the customer his change. Does the answer suggest a general result about giving change? Resolve the problem if a 20¢ piece (in addition to other United States coins) were available.

6 A company needs to have a working machine during each of the next six years. Currently, it has a new machine. At the beginning of each year, the company may keep the machine or sell it and buy a new one. A machine cannot be kept for more than three years. A new machine costs $5,000. The revenues earned by a machine, the cost of maintaining it, and the salvage value that can be obtained by selling it at the end of a year depend on the age of the machine (see Table 21). Use dynamic programming to maximize the net profit earned during the next six years.

7 A company needs the following number of workers during each of the next five years: year 1, 15; year 2, 30; year 3, 10; year 4, 30; year 5, 20. At present, the company has 20 workers. Each worker is paid $30,000 per year. At the beginning of each year, workers may be hired or fired. It costs $10,000 to hire a worker and $20,000 to fire a worker. A newly hired worker can be used to meet the current year's worker requirement. During each year, 10% of all workers quit (workers who quit do not incur any firing cost).

a With dynamic programming, formulate a recursion that can be used to minimize the total cost incurred in meeting the worker requirements of the next five years.

b How would the recursion be modified if hired workers cannot be used to meet worker requirements until the year following the year in which they are hired?

8 At the beginning of each year, Barnes Carr Oil sets the world oil price. If a price p is set, then $D(p)$ barrels of oil will be demanded by world customers. We assume that during any year, each oil company sells the same number of barrels of oil. It costs Barnes Carr Oil c dollars to extract and refine each barrel of oil. Barnes Carr cannot set too high a price, however, because if a price p is set and there are currently N oil companies, then $g(p, N)$ oil companies will enter the oil business [$g(p, N)$ could be negative]. Setting too high a price will dilute future profits because of the entrance of new companies. Barnes Carr wants to maximize the discounted profit the company will earn over the next 20 years. Formulate a recursion that will aid Barnes Carr in meeting its goal. Initially, there are 10 oil companies.

9 For a computer to work properly, three subsystems of the computer must all function properly. To increase the reliability of the computer, spare units may be added to each system. It costs $100 to add a spare unit to system 1, $300 to system 2, and $200 to system 3. As a function of the number of added spares (a maximum of two spares may be added to each system), the probability that each system will work is given in Table 22. Use dynamic programming to maximize the probability that the computer will work properly, given that $600 is available for spare units.

Group B

10 During any year, I can consume any amount that does not exceed my current wealth. If I consume c dollars during a year, I earn c^a units of happiness. By the beginning of the next year, the previous year's ending wealth grows by a factor k.

a Formulate a recursion that can be used to maximize total utility earned during the next T years. Assume I originally have w_0 dollars.

b Let $f_t(w)$ be the maximum utility earned during years $t, t + 1, \ldots, T$, given that I have w dollars at the beginning of year t; and $c_t(w)$ be the amount that should be consumed during year t to attain $f_t(w)$. By working backward, show that for appropriately chosen constants a_t and b_t,

$$f_t(w) = b_t w^a \quad \text{and} \quad c_t(w) = a_t w$$

Interpret these results.

11 At the beginning of month t, farmer Smith has x_t bushels of wheat in his warehouse. He has the opportunity to sell wheat at a price s_t dollars per bushel and can buy wheat at p_t dollars per bushel. Farmer Smith's warehouse can hold at most C units at the end of each month.

a Formulate a recursion that can be used to maximize the total profit earned during the next T months.

b Let $f_t(x_t)$ be the maximum profit that can be earned during months $t, t + 1, \ldots, T$, given that x_t bushels of

TABLE 21

	Age of Machine at Beginning of Year 0 Year	1 Year	2 Years
Revenues ($)	4,500	3,000	1,500
Operating Costs ($)	500	700	1,100
Salvage Value at End of Year ($)	3,000	1,800	500

TABLE 22

Number of Spares	Probability That a System Works System 1	System 2	System 3
0	.85	.60	.70
1	.90	.85	.90
2	.95	.95	.98

wheat are in the warehouse at the beginning of month t. By working backward, show that for appropriately chosen constants a_t and b_t,

$$f_t(x_t) = a_t + b_t x_t$$

c During any given month, show that the profit-maximizing policy has the following properties: (1) The amount sold during month t will equal either x_t or zero. (2) The amount purchased during a given month will be either zero or sufficient to bring the month's ending stock to C bushels.

REFERENCES

The following references are oriented toward applications and are written at an intermediate level:

Dreyfus, S., and A. Law. *The Art and Theory of Dynamic Programming.* Orlando, Fla.: Academic Press, 1977.

Nemhauser, G. *Introduction to Dynamic Programming*. New York: Wiley, 1966.

Wagner, H. *Principles of Operations Research,* 2d ed. Englewood Cliffs, N.J.: Prentice Hall, 1975.

The following five references are oriented toward theory and are written at a more advanced level:

Bellman, R. *Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1957.

Bellman, R., and S. Dreyfus. *Applied Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1962.

Bersetkas, D. *Dynamic Programming and Optimal Control,* vol. 1. Cambridge, Mass.: Athena Scientific, 2000.

Denardo, E. *Dynamic Programming: Theory and Applications.* Englewood Cliffs, N.J.: Prentice Hall, 1982.

Whittle, P. *Optimization Over Time: Dynamic Programming and Stochastic Control,* vol. 1. New York: Wiley, 1982.

Morton, T. "Planning Horizons for Dynamic Programs," *Operations Research* 27(1979):730–743. A discussion of turnpike theorems.

Peterson, R., and E. Silver. *Decision Systems for Inventory Management and Production Planning.* New York: Wiley, 1998. Discusses the Silver–Meal method.

Waddell, R. "A Model for Equipment Replacement Decisions and Policies," *Interfaces* 13(1983):1–8. An application of the equipment replacement model.

Wagner, H., and T. Whitin. "Dynamic Version of the Economic Lot Size Model," *Management Science* 5(1958):89–96. Discusses Wagner–Whitin method.

Probabilistic Dynamic Programming

Recall from our study of deterministic dynamic programming that many recursions were of the following form:

$$f_t\text{ (current state)} = \min_{\substack{\text{all feasible}\\ \text{decisions}}} \text{(or max)}\{\text{costs during current stage} + f_{t+1}\text{ (new state)}\}$$

For all the examples in Chapter 18, a specification of the current state and current decision was enough to tell us *with certainty* the new state and the costs during the current stage. In many practical problems, these factors may not be known with certainty, even if the current state and decision are known. For example, in the inventory model of Section 18.3, we assumed that each period's demand was known at the beginning of the problem. In most situations, it would be more realistic to assume that period t's demand is a random variable whose value is not known until after period t's production decision is made. Even if we know the current period's state (beginning inventory level) and decision (production during the current period), the next period's state and the current period's cost will be random variables whose values are not known until the value of period t's demand is known. The Chapter 18 discussion simply does not apply to this problem.

In this chapter, we explain how to use dynamic programming to solve problems in which the current period's cost or the next period's state are random. We call these problems *probabilistic dynamic programming problems* (or PDPs). In a PDP, the decision maker's goal is usually to minimize expected (or expected discounted) cost incurred or to maximize expected (or expected discounted) reward earned over a given time horizon. Chapter 19 concludes with a brief study of *Markov decision processes.* A Markov decision process is just a probabilistic dynamic programming problem in which the decision maker faces an infinite horizon.

19.1 When Current Stage Costs Are Uncertain, but the Next Period's State Is Certain

For problems in this section, the next period's state is known with certainty, but the reward earned during the current stage is not known with certainty (given the current state and decision).

EXAMPLE 1

For a price of \$1/gallon, the Safeco Supermarket chain has purchased 6 gallons of milk from a local dairy. Each gallon of milk is sold in the chain's three stores for \$2/gallon. The dairy must buy back for 50¢/gallon any milk that is left at the end of the day. Unfortunately for Safeco, demand for each of the chain's three stores is uncertain. Past data indicate that the daily demand at each store is as shown in Table 1. Safeco wants to allo-

TABLE 1
Probability Distributions for Daily Milk Demand

	Daily Demand (gallons)	Probability
Store 1	1	.60
	2	0
	3	.40
Store 2	1	.50
	2	.10
	3	.40
Store 3	1	.40
	2	.30
	3	.30

cate the 6 gallons of milk to the three stores so as to maximize the expected net daily profit (revenues less costs) earned from milk. Use dynamic programming to determine how Safeco should allocate the 6 gallons of milk among the three stores.

Solution With the exception of the fact that the demand (and therefore the revenue) is uncertain, this problem is very similar to the resource allocation problems studied in Section 18.4.

Observe that since Safeco's daily purchase costs are always \$6, we may concentrate our attention on the problem of allocating the milk to maximize daily expected revenue earned from the 6 gallons.

Define

$r_t(g_t)$ = expected revenue earned from g_t gallons assigned to store t

$f_t(x)$ = maximum expected revenue earned from x gallons assigned to stores $t, t + 1, \ldots, 3$

Since $f_3(x)$ must by definition be the expected revenue earned from assigning x gallons of milk to store 3, we see that $f_3(x) = r_3(x)$. For $t = 1, 2$, we may write

$$f_t(x) = \max_{g_t} \{r_t(g_t) + f_{t+1}(x - g_t)\} \tag{1}$$

where g_t must be a member of $\{0, 1, \ldots, x\}$. Equation (1) follows, because for any choice of g_t (the number of gallons assigned to store t), the expected revenue earned from store $t, t + 1, \ldots, 3$ will be the sum of the expected revenue earned from store t if g_t gallons are assigned to store t plus the maximum expected revenue that can be earned from the stores $t + 1, t + 2, \ldots, 3$ when $x - g_t$ gallons are assigned to these stores. To compute the optimal allocation of milk to the stores, we begin by computing $f_3(0), f_3(1), \ldots, f_3(6)$. Then we use Equation (1) to compute $f_2(0), f_2(1), \ldots, f_2(6)$. Finally we determine $f_1(6)$.

We begin by computing the $r_t(g_t)$'s. Note that it would be foolish to assign more than 3 gallons to any store. For this reason, we compute the $r_t(g_t)$'s only for $g_t = 0, 1, 2,$ or 3. As an example, we compute $r_3(2)$, the expected revenue earned if 2 gallons are assigned to store 3. If the demand at store 3 is for 2 or more gallons, both gallons assigned to store 3 will be sold, and \$4 in revenue will be earned. If the demand at store 3 is 1 gallon, 1 gallon will be sold for \$2, and 1 gallon will be returned for 50¢. Hence, if demand at store 3 is for 1 gallon, a revenue of \$2.50 will be earned. Since there is a .60 chance that demand at store 3 will be for 2 or more gallons and a .40 chance that store 3 demand will be for 1 gallon, it follows that $r_3(2) = (.30 + .30)(4.00) + .40(2.50) = \$3.40$. Similar computations yield the following results:

$$\begin{array}{lll} r_3(0) = \$0 & r_2(0) = \$0 & r_1(0) = \$0 \\ r_3(1) = \$2.00 & r_2(1) = \$2.00 & r_1(1) = \$2.00 \\ r_3(2) = \$3.40 & r_2(2) = \$3.25 & r_1(2) = \$3.10 \\ r_3(3) = \$4.35 & r_2(3) = \$4.35 & r_1(3) = \$4.20 \end{array}$$

We now use (1) to determine an optimal allocation of milk to stores. Let $g_t(x)$ be an allocation of milk to store t that attains $f_t(x)$. Then

$$\begin{array}{ll} f_3(0) = r_3(0) = 0 & g_3(0) = 0 \\ f_3(1) = r_3(1) = 2.00 & g_3(1) = 1 \\ f_3(2) = r_3(2) = 3.40 & g_3(2) = 2 \\ f_3(3) = r_3(3) = 4.35 & g_3(3) = 3 \end{array}$$

We need not compute $f_3(4)$, $f_3(5)$, and $f_3(6)$, because an optimal allocation will never have more than 3 gallons to allocate to a single store (demand at any store is never more than 3 gallons).

Using (1) to work backward, we obtain

$$f_2(0) = r_2(0) + f_3(0 - 0) = 0 \qquad g_2(0) = 0$$

$$f_2(1) = \max \begin{cases} r_2(0) + f_3(1 - 0) = 2.00^* \\ r_2(1) + f_3(1 - 1) = 2.00^* \end{cases} \qquad g_2(1) = 0 \text{ or } 1$$

$$f_2(2) = \max \begin{cases} r_2(0) + f_3(2 - 0) = 0 + 3.40 = 3.40 \\ r_2(1) + f_3(2 - 1) = 2.00 + 2.00 = 4.00^* \\ r_2(2) + f_3(2 - 2) = 3.25 + 0 = 3.25 \end{cases} \qquad g_2(2) = 1$$

$$f_2(3) = \max \begin{cases} r_2(0) + f_3(3 - 0) = 0 + 4.35 = 4.35 \\ r_2(1) + f_3(3 - 1) = 2.00 + 3.40 = 5.40^* \\ r_2(2) + f_3(3 - 2) = 3.25 + 2.00 = 5.25 \\ r_2(3) + f_3(3 - 3) = 4.35 + 0 = 4.35 \end{cases} \qquad g_2(3) = 1$$

Note that in computing $f_2(4)$, $f_2(5)$, and $f_2(6)$, we need not consider any allocation for more than 3 gallons to store 2 or any that leaves more than 3 gallons for store 3.

$$f_2(4) = \max \begin{cases} r_2(1) + f_3(4 - 1) = 2.00 + 4.35 = 6.35 \\ r_2(2) + f_3(4 - 2) = 3.25 + 3.40 = 6.65^* \\ r_2(3) + f_3(4 - 3) = 4.35 + 2.00 = 6.35 \end{cases} \qquad g_2(4) = 2$$

$$f_2(5) = \max \begin{cases} r_2(2) + f_3(5 - 2) = 3.25 + 4.35 = 7.60 \\ r_2(3) + f_3(5 - 3) = 4.35 + 3.40 = 7.75^* \end{cases} \qquad g_2(5) = 3$$

$$f_2(6) = r_2(3) + f_3(6 - 3) = 4.35 + 4.35 = 8.70^* \qquad g_2(6) = 3$$

Finally,

$$f_1(6) = \max \begin{cases} r_1(0) + f_2(6 - 0) = 0 + 8.70 \\ r_1(1) + f_2(6 - 1) = 2.00 + 7.75 = 9.75^* \\ r_1(2) + f_2(6 - 2) = 3.10 + 6.65 = 9.75^* \\ r_1(3) + f_2(6 - 3) = 4.20 + 5.40 = 9.60 \end{cases} \qquad g_1(6) = 1 \text{ or } 2$$

Thus, we can either assign 1 or 2 gallons to store 1. Suppose we arbitrarily choose to assign 1 gallon to store 1. Then we have $6 - 1 = 5$ gallons for stores 2 and 3. Since $f_2(5)$

is attained by $g_2(5) = 3$, we assign 3 gallons to store 2. Then $5 - 3 = 2$ gallons are available for store 3. Since $g_3(2) = 2$, we assign 2 gallons to store 3. Note that although this policy obtains the maximum expected revenue, $f_1(6) = \$9.75$, the total revenue actually received on a given day may be more or less than \$9.75. For example, if demand at each store were 1 gallon, total revenue would be $3(2.00) + 3(0.50) = \$7.50$, whereas if demand at each store were 3 gallons, all the milk would be sold at \$2/gallon, and the total revenue would be $6(2.00) = \$12.00$.

PROBLEMS

Group A

1 In Example 1, find another allocation of milk that maximizes expected daily revenue.

2 Suppose that \$4 million is available for investment in three projects. The probability distribution of the net present value earned from each project depends on how much is invested in each project. Let $\mathbf{I}_t$ be the random variable denoting the net present value earned by project t. The distribution of $\mathbf{I}_t$ depends on the amount of money invested in project t, as shown in Table 2 (a zero investment in a project always earns a zero NPV). Use dynamic programming to determine an investment allocation that maximizes the expected NPV obtained from the three investments.

TABLE 2
Investment Probability for Problem 2

	Investment (millions)	Probability		
Project 1	\$1	$P(\mathbf{I}_1 = 2) = .6$	$P(\mathbf{I}_1 = 4) = .3$	$P(\mathbf{I}_1 = 5) = .1$
	\$2	$P(\mathbf{I}_1 = 4) = .5$	$P(\mathbf{I}_1 = 6) = .3$	$P(\mathbf{I}_1 = 8) = .2$
	\$3	$P(\mathbf{I}_1 = 6) = .4$	$P(\mathbf{I}_1 = 7) = .5$	$P(\mathbf{I}_1 = 10) = .1$
	\$4	$P(\mathbf{I}_1 = 7) = .2$	$P(\mathbf{I}_1 = 9) = .4$	$P(\mathbf{I}_1 = 10) = .4$
Project 2	\$1	$P(\mathbf{I}_2 = 1) = .5$	$P(\mathbf{I}_2 = 2) = .4$	$P(\mathbf{I}_2 = 4) = .1$
	\$2	$P(\mathbf{I}_2 = 3) = .4$	$P(\mathbf{I}_2 = 5) = .4$	$P(\mathbf{I}_2 = 6) = .2$
	\$3	$P(\mathbf{I}_2 = 4) = .3$	$P(\mathbf{I}_2 = 6) = .3$	$P(\mathbf{I}_2 = 8) = .4$
	\$4	$P(\mathbf{I}_2 = 3) = .4$	$P(\mathbf{I}_2 = 8) = .3$	$P(\mathbf{I}_2 = 9) = .3$
Project 3	\$1	$P(\mathbf{I}_3 = 0) = .2$	$P(\mathbf{I}_3 = 4) = .6$	$P(\mathbf{I}_3 = 5) = .2$
	\$2	$P(\mathbf{I}_3 = 4) = .4$	$P(\mathbf{I}_3 = 6) = .4$	$P(\mathbf{I}_3 = 7) = .2$
	\$3	$P(\mathbf{I}_3 = 5) = .3$	$P(\mathbf{I}_3 = 7) = .4$	$P(\mathbf{I}_3 = 8) = .3$
	\$4	$P(\mathbf{I}_3 = 6) = .1$	$P(\mathbf{I}_3 = 8) = .5$	$P(\mathbf{I}_3 = 9) = .4$

19.2 A Probabilistic Inventory Model

In this section, we modify the inventory model of Section 18.3 to allow for uncertain demand. This will illustrate the difficulties involved in solving a PDP for which the state during the next period is uncertain (given the current state and current decision).

EXAMPLE 2 Three-Period Production Policy

Consider the following three-period inventory problem. At the beginning of each period, a firm must determine how many units should be produced during the current period. During a period in which x units are produced, a production cost $c(x)$ is incurred, where $c(0) = 0$, and for $x > 0$, $c(x) = 3 + 2x$. Production during each period is limited to at most 4 units. After production occurs, the period's random demand is observed. Each period's demand is equally likely to be 1 or 2 units. After meeting the current period's demand out of current production and inventory, the firm's end-of-period inventory is evaluated, and a holding cost of \$1 per unit is assessed. Because of limited capacity, the inventory at the end of each period cannot exceed 3 units. It is required that all demand be met on time. Any inventory on hand at the end of period 3 can be sold at \$2 per unit. At the beginning of period 1, the firm has 1 unit of inventory. Use dynamic programming to determine a production policy that minimizes the expected net cost incurred during the three periods.

Solution Define $f_t(i)$ to be the minimum expected net cost incurred during the periods $t, t+1, \ldots, 3$ when the inventory at the beginning of period t is i units. Then

$$f_3(i) = \min_x \{c(x) + (\tfrac{1}{2})(i + x - 1) + (\tfrac{1}{2})(i + x - 2) - (\tfrac{1}{2})2(i + x - 1) - (\tfrac{1}{2})2(i + x - 2)\} \tag{2}$$

where x must be a member of $\{0, 1, 2, 3, 4\}$ and x must satisfy $(2 - i) \le x \le (4 - i)$.

Equation (2) follows, because if x units are produced during period 3, the net cost during period 3 is (expected production cost) + (expected holding cost) − (expected salvage value). If x units are produced, the expected production cost is $c(x)$, and there is a $\frac{1}{2}$ chance that the period 3 holding cost will be $i + x - 1$ and a $\frac{1}{2}$ chance that it will be $i + x - 2$. Hence, the period 3 expected holding cost will be $(\frac{1}{2})(i + x - 1) + (\frac{1}{2})(i + x - 2) = i + x - \frac{3}{2}$. Similar reasoning shows that the expected salvage value (a negative cost) at the end of period 3 will be $(\frac{1}{2})2(i + x - 1) + (\frac{1}{2})2(i + x - 2) = 2i + 2x - 3$. To ensure that period 3 demand is met, we must have $i + x \ge 2$, or $x \ge 2 - i$. Similarly, to ensure that ending period three inventory does not exceed 3 units, we must have $i + x - 1 \le 3$, or $x \le 4 - i$.

For $t = 1, 2$, we can derive the recursive relation for $f_t(i)$ by noting that for any month t production level x, the expected costs incurred during periods $t, t + 1, \ldots, 3$ are the sum of the expected costs incurred during period t and the expected costs incurred during periods $t + 1, t + 2, \ldots, 3$. As before, if x units are produced during month t, the expected cost during month t will be $c(x) + (\frac{1}{2})(i + x - 1) + (\frac{1}{2})(i + x - 2)$. (Note that during periods 1 and 2, no salvage value is received.) If x units are produced during month t, the expected cost during periods $t + 1, t + 2, \ldots, 3$ is computed as follows. Half of the time, the demand during period t will be 1 unit, and the inventory at the beginning of period $t + 1$ will be $i + x - 1$. In this situation, the expected costs incurred during periods $t + 1, t + 2, \ldots, 3$ (assuming we act optimally during these periods) is $f_{t+1}(i + x - 1)$. Similarly, there is a $\frac{1}{2}$ chance that the inventory at the beginning of period $t + 1$ will be $i + x - 2$. In this case, the expected cost incurred during periods $t + 1, t + 2, \ldots, 3$ will be $f_{t+1}(i + x - 2)$. In summary, the expected cost during periods $t + 1, t + 2, \ldots, 3$ will be $(\frac{1}{2})f_{t+1}(i + x - 1) + (\frac{1}{2})f_{t+1}(i + x - 2)$. With this in mind, we may write for $t = 1, 2$,

$$f_t(i) = \min_x [c(x) + (\tfrac{1}{2})(i + x - 1) + (\tfrac{1}{2})(i + x - 2) + (\tfrac{1}{2})f_{t+1}(i + x - 1) + (\tfrac{1}{2})f_{t+1}(i + x - 2)] \tag{3}$$

where x must be a member of $\{0, 1, 2, 3, 4\}$ and x must satisfy $(2 - i) \le x \le (4 - i)$.

Generalizing the reasoning that led to (3) yields the following important observation concerning the formulation of PDPs. Suppose the possible states during period $t + 1$ are $s_1, s_2, \ldots, s_n$ and the probability that the period $t + 1$ state will be s_i is p_i. Then the minimum expected cost incurred during periods $t + 1, t + 2, \ldots,$ end of the problem is

$$\sum_{i=1}^{i=n} p_i f_{t+1}(s_i)$$

where $f_{t+1}(s_i)$ is the minimum expected cost incurred from period $t + 1$ to the end of the problem, given that the state during period $t + 1$ is s_i.

We define $x_t(i)$ to be a period t production level attaining the minimum in (3) for $f_t(i)$. We now work backward until $f_1(1)$ is determined. The relevant computations are summarized in Tables 3, 4, and 5. Since each period's ending inventory must be nonnegative and cannot exceed 3 units, the state during each period must be 0, 1, 2, or 3.

As in Section 18.3, we begin by producing $x_1(1) = 3$ units during period 1. We cannot, however, determine period 2's production level until period 1's demand is observed. Also,

TABLE 3
Computations for $f_3(i)$

i	x	$c(x)$	Expected Holding Cost $(i + x - \frac{3}{2})$	Expected Salvage Value $(2i + 2x - 3)$	Total Expected Cost	$f_3(i)$ $x_3(i)$
3	0	0	$\frac{3}{2}$	3	$-\frac{3}{2}$*	$f_3(3) = -\frac{3}{2}$
3	1	5	$\frac{5}{2}$	5	$\frac{5}{2}$	$x_3(3) = 0$
2	0	0	$\frac{1}{2}$	1	$-\frac{1}{2}$*	$f_3(2) = -\frac{1}{2}$
2	1	5	$\frac{3}{2}$	3	$\frac{7}{2}$	$x_3(2) = 0$
2	2	7	$\frac{5}{2}$	5	$\frac{9}{2}$	
1	1	5	$\frac{1}{2}$	1	$\frac{9}{2}$*	$f_3(1) = \frac{9}{2}$
1	2	7	$\frac{3}{2}$	3	$\frac{11}{2}$	$x_3(1) = 1$
1	3	9	$\frac{5}{2}$	5	$\frac{13}{2}$	
0	2	7	$\frac{1}{2}$	1	$\frac{13}{2}$*	$f_3(0) = \frac{13}{2}$
0	3	9	$\frac{3}{2}$	3	$\frac{15}{2}$	$x_3(0) = 2$
0	4	11	$\frac{5}{2}$	5	$\frac{17}{2}$	

TABLE 4
Computations for $f_2(i)$

i	x	$c(x)$	Expected Holding Cost $(i + x - \frac{3}{2})$	Expected Future Cost $((\frac{1}{2})f_3(i + x - 1)$ $+(\frac{1}{2})f_3\,(i + x - 2))$	Total Expected Cost Periods 2,3	$f_2(i)$ $x_2(i)$
3	0	0	$\frac{3}{2}$	2	$\frac{7}{2}$*	$f_2(3) = \frac{7}{2}$
3	1	5	$\frac{5}{2}$	-1	$\frac{13}{2}$	$x_2(3) = 0$
2	0	0	$\frac{1}{2}$	$\frac{11}{2}$	6*	$f_2(2) = 6$
2	1	5	$\frac{3}{2}$	2	$\frac{17}{2}$	$x_2(2) = 0$
2	2	7	$\frac{5}{2}$	-1	$\frac{17}{2}$	
1	1	5	$\frac{1}{2}$	$\frac{11}{2}$	11	$f_2(1) = \frac{21}{2}$
1	2	7	$\frac{3}{2}$	2	$\frac{21}{2}$*	$x_2(1) = 2$ or 3
1	3	9	$\frac{5}{2}$	-1	$\frac{21}{2}$*	
0	2	7	$\frac{1}{2}$	$\frac{11}{2}$	13	$f_2(0) = \frac{25}{2}$
0	3	9	$\frac{3}{2}$	2	$\frac{25}{2}$*	$x_2(0) = 3$ or 4
0	4	11	$\frac{5}{2}$	-1	$\frac{25}{2}$*	

TABLE 5
Computations for $f_1(1)$

x	$c(x)$	Expected Holding Cost $(i + x - \frac{3}{2})$	Expected Future Cost $((\frac{1}{2})f_2(i + x - 1) + (\frac{1}{2})f_2(i + x - 2))$	Total Expected Cost Periods 1–3	$f_1(1)$ $x_1(1)$
1	5	$\frac{1}{2}$	$\frac{23}{2}$	17	$f_1(1) = \frac{65}{4}$
2	7	$\frac{3}{2}$	$\frac{33}{4}$	$\frac{67}{4}$	$x_1(1) = 3$
3	9	$\frac{5}{2}$	$\frac{19}{4}$	$\frac{65}{4}$*	

period 3's production level cannot be determined until period 2's demand is observed. To illustrate the idea, we determine the optimal production schedule if period 1 and period 2 demands are both 2 units. Since $x_1(1) = 3$, 3 units will be produced during period 1. Then period 2 will begin with an inventory of $1 + 3 - 2 = 2$ units, so $x_2(2) = 0$ units should be produced. After period 2's demand of 2 units is met, period 3 will begin with $2 - 2 = 0$ units on hand. Thus, $x_3(0) = 2$ units will be produced during period 3.

In contrast, suppose that period 1 and period 2 demands are both 1 unit. As before, $x_1(1) = 3$ units will be produced during period 1. Then period 2 will begin with $1 + 3 - 1 = 3$ units, and $x_2(3) = 0$ units will be produced during period 2. Then period 3 will begin with $3 - 1 = 2$ units on hand, and $x_3(2) = 0$ units will be produced during period 3. Note that the optimal production policy has adapted to the low demand by reducing period 3 production. This example illustrates an important aspect of dynamic programming solutions for problems in which future states are not known with certainty at the beginning of the problem: *If a random factor (such as random demand) influences transitions from the period t state to the period t + 1 state, the optimal action for period t cannot be determined until period t's state is known.*

(s, S) Policies

Consider the following modification of the dynamic lot-size model of Section 18.7, for which there exists an optimal production policy called an (s, S) inventory policy:

1 The cost of producing $x > 0$ units during a period consists of a fixed cost K and a per-unit variable production cost c.

2 With a probability $p(x)$, the demand during a given period will be x.

3 A holding cost of h per unit is assessed on each period's ending inventory. If we are short, a per-unit shortage cost of d is incurred. (The case where no shortages are allowed may be obtained by letting d be very large.)

4 The goal is to minimize the total expected cost incurred during periods $1, 2, \ldots, T$.

5 All demands must be met by the end of period T.

For such an inventory problem, Scarf (1960) used dynamic programming to prove that there exists an optimal production policy of the following form: For each t ($t = 1, 2, \ldots, T$) there exists a pair of numbers (s_t, S_t) such that if i_{t-1}, the entering inventory for period

t, is less than s_t, then an amount $S_t - i_{t-1}$ is produced; if $i_{t-1} \geq s_t$, then it is optimal not to produce during period t. Such a policy is called an **(*s*, *S*) policy.**

For Example 2, our calculations show that $s_2 = 2$, $S_2 = 3$ or 4, $s_3 = 2$, and $S_3 = 2$. Thus, if we enter period 2 with 1 or 0 units, we produce enough to bring our stock level (before meeting period 2 demand) up to 3 or 4 units. If we enter period 2 with more than 1 unit, then no production should take place during period 2.

PROBLEMS

Group A

1 For Example 2, suppose that the period 1 demand is 1 unit, and the period 2 demand is 2 units. What would be the optimal production schedule?

2 Re-solve Example 2 if the end-of-period holding cost is \$2 per unit.

3 In Example 2, suppose that shortages are allowed, and each shortage results in a lost sale and a cost incurred of \$3. Now re-solve Example 2.

Group B

4 Chip Bilton sells sweatshirts at State U football games. He is equally likely to sell 200 or 400 sweatshirts at each game. Each time Chip places an order, he pays \$500 plus \$5 for each sweatshirt he orders. Each sweatshirt sells for \$8. A holding cost of \$2 per shirt (because of the opportunity cost for capital tied up in sweatshirts as well as storage costs) is assessed against each shirt left at the end of a game. Chip can store at most 400 shirts after each game. Assuming that the number of shirts ordered by Chip must be a multiple of 100, determine an ordering policy that maximizes expected profits earned during the first three games of the season. Assume that any leftover sweatshirts have a value of \$6.

19.3 How to Maximize the Probability of a Favorable Event Occurring[†]

There are many occasions on which the decision maker's goal is to maximize the probability of a favorable event occurring. For instance, a company may want to maximize its probability of reaching a specified level of annual profits. To solve such a problem, we assign a reward of 1 if the favorable event occurs and a reward of 0 if it does not occur. Then the maximization of expected reward will be equivalent to maximizing the probability that the favorable event will occur. Also, the maximum expected reward will equal the maximum probability of the favorable event occurring. The following two examples illustrate how this idea may be used to solve some fairly complex problems.

EXAMPLE 3 Gambling Game

A gambler has \$2. She is allowed to play a game of chance four times, and her goal is to maximize her probability of ending up with a least \$6. If the gambler bets b dollars on a play of the game, then with probability .40, she wins the game and increases her capital position by b dollars; with probability .60, she loses the game and decreases her capital by b dollars. On any play of the game, the gambler may not bet more money than she has available. Determine a betting strategy that will maximize the gambler's probability of attaining a wealth of at least \$6 by the end of the fourth game. We assume that bets of zero dollars (that is, not betting) are permissible.

Solution Define $f_t(d)$ to be the probability that by the end of game 4, the gambler will have at least

[†]This section covers topics that may be omitted with no loss of continuity.

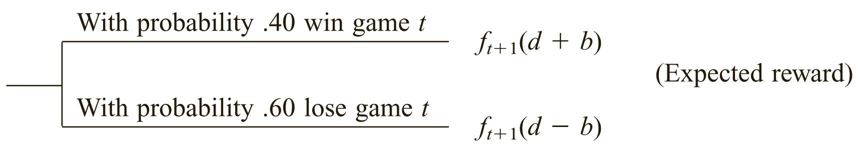
\$6, given that she acts optimally and has d dollars immediately before the game is played for the tth time. If we give the gambler a reward of 1 when her ending wealth is at least \$6 and a reward of 0 if it is less, then $f_t(d)$ will equal the maximum expected reward that can be earned during games $t, t+1, \ldots, 4$ if the gambler has d dollars immediately before the tth play of the game. As usual, we define $b_t(d)$ dollars to be a bet size that attains $f_t(d)$.

If the gambler is playing the game for the fourth and final time, her optimal strategy is clear: If she has \$6 or more, don't bet anything, but if she has less than \$6, bet enough money to ensure (if possible) that she will have \$6 if she wins the last game. Note that if she begins game 4 with \$0, \$1, or \$2, there is no way to win (no way to earn a reward of 1). This reasoning yields the following results:

$f_4(0) = 0$	$b_4(0) = \$0$
$f_4(1) = 0$	$b_4(1) = \$0$ or $\$1$
$f_4(2) = 0$	$b_4(2) = \$0, \$1,$ or $\$2$
$f_4(3) = .40$	$b_4(3) = \$3$
$f_4(4) = .40$	$b_4(4) = \$2, \$3,$ or $\$4$
$f_4(5) = .40$	$b_4(5) = \$1, \$2, \$3, \$4,$ or $\$5$

For $d \geq 6$,

$$f_4(d) = 1 \qquad b_4(d) = \$0, \$1, \ldots, \$(d-6)$$


With probability .40 win game t
$f_{t+1}(d+b)$
(Expected reward)
With probability .60 lose game t
$f_{t+1}(d-b)$

For $t \leq 3$, we can find a recursion for $f_t(d)$ by noting that if the gambler has d dollars, is about to play the game for the tth time, and bets b dollars, then the following diagram summarizes what can occur:
Thus, if the gambler has d dollars at the beginning of game t and bets b dollars, the expected reward (or expected probability of reaching \$6) will be $.4f_{t+1}(d+b) + .6f_{t+1}(d-b)$. This leads to the following recursion:

$$f_t(d) = \max_b (.4f_{t+1}(d+b) + .6f_{t+1}(d-b)) \tag{4}$$

where b must be a member of $\{0, 1, \ldots, d\}$. Then $b_t(d)$ is any bet size that attains the maximum in (4) for $f_t(d)$. Using (4), we work backward until $f_1(2)$ has been determined.

Stage 3 Computations

$$f_3(0) = 0 \qquad b_3(0) = \$0$$

$$f_3(1) = \max \begin{cases} .4f_4(1) + .6f_4(1) = 0^* & \text{(Bet \$0)} \\ .4f_4(2) + .6f_4(0) = 0^* & \text{(Bet \$1)} \end{cases}$$

Thus, $f_3(1) = 0$, and $b_3(1) = \$0$ or $\$1$.

$$f_3(2) = \max \begin{cases} .4f_4(2) + .6f_4(2) = 0 & \text{(Bet \$0)} \\ .4f_4(3) + .6f_4(1) = .16^* & \text{(Bet \$1)} \\ .4f_4(4) + .6f_4(0) = .16^* & \text{(Bet \$2)} \end{cases}$$

Thus, $f_3(2) = .16$, and $b_3(2) = \$1$ or $\$2$.

$$f_3(3) = \max \left\{ \vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}} \right.$$

Thus, $f_3(3) = .40$, and $b_3(3) = \$0$ or $\$3$.

$$f_4(4) = \max \begin{cases} .4f_4(4) + .6f_4(4) = .40^* & \text{(Bet \$0)} \\ .4f_4(5) + .6f_4(3) = .40^* & \text{(Bet \$1)} \\ .4f_4(6) + .6f_4(2) = .40^* & \text{(Bet \$2)} \\ .4f_4(7) + .6f_4(1) = .40^* & \text{(Bet \$3)} \\ .4f_4(8) + .6f_4(0) = .40^* & \text{(Bet \$4)} \end{cases}$$

Thus, $f_3(4) = .40$, and $b_3(4) = \$0, \$1, \$2, \$3$, or $\$4$.

$$f_3(5) = \max \begin{cases} .4f_4(5) + .6f_4(5) \ \ = .40 & \text{(Bet \$0)} \\ .4f_4(6) + .6f_4(4) \ \ = .64^* & \text{(Bet \$1)} \\ .4f_4(7) + .6f_4(3) \ \ = .64^* & \text{(Bet \$2)} \\ .4f_4(8) + .6f_4(2) \ \ = .40 & \text{(Bet \$3)} \\ .4f_4(9) + .6f_4(1) \ \ = .40 & \text{(Bet \$4)} \\ .4f_4(10) + .6f_4(0) = .40 & \text{(Bet \$5)} \end{cases}$$

Thus, $f_3(5) = .64$, and $b_3(5) = \$1$ or $\$2$. For $d \geq 6$, $f_3(d) = 1$, and $b_3(d) = \$0, \$1, \ldots, \$(d - 6)$.

Stage 2 Computations

$$f_2(0) = 0 \qquad b_2(0) = \$0$$

$$f_2(1) = \max \begin{cases} .4f_3(1) + .6f_3(1) = 0 & \text{(Bet \$0)} \\ .4f_3(2) + .6f_3(0) = .064^* & \text{(Bet \$1)} \end{cases}$$

Thus, $f_2(1) = .064$, and $b_2(1) = \$1$.

$$f_2(2) = \max \begin{cases} .4f_3(2) + .6f_3(2) = .16^* & \text{(Bet \$0)} \\ .4f_3(3) + .6f_3(1) = .16^* & \text{(Bet \$1)} \\ .4f_3(4) + .6f_3(0) = .16^* & \text{(Bet \$2)} \end{cases}$$

Thus, $f_2(2) = .16$, and $b_2(2) = \$0, \$1$, or $\$2$.

$$f_2(3) = \max \begin{cases} .4f_3(3) + .6f_3(3) = .40^* & \text{(Bet \$0)} \\ .4f_3(4) + .6f_3(2) = .256 & \text{(Bet \$1)} \\ .4f_3(5) + .6f_3(1) = .256 & \text{(Bet \$2)} \\ .4f_3(6) + .6f_3(0) = .40^* & \text{(Bet \$3)} \end{cases}$$

Thus, $f_2(3) = .40$, and $b_2(3) = \$0$ or $\$3$.

$$f_2(4) = \max \begin{cases} .4f_3(4) + .6f_3(4) = .40 & \text{(Bet \$0)} \\ .4f_3(5) + .6f_3(3) = .496^* & \text{(Bet \$1)} \\ .4f_3(6) + .6f_3(2) = .496^* & \text{(Bet \$2)} \\ .4f_3(7) + .6f_3(1) = .40 & \text{(Bet \$3)} \\ .4f_3(8) + .6f_3(0) = .40 & \text{(Bet \$4)} \end{cases}$$

Thus, $f_2(4) = .496$, and $b_2(4) = \$1$ or $\$2$.

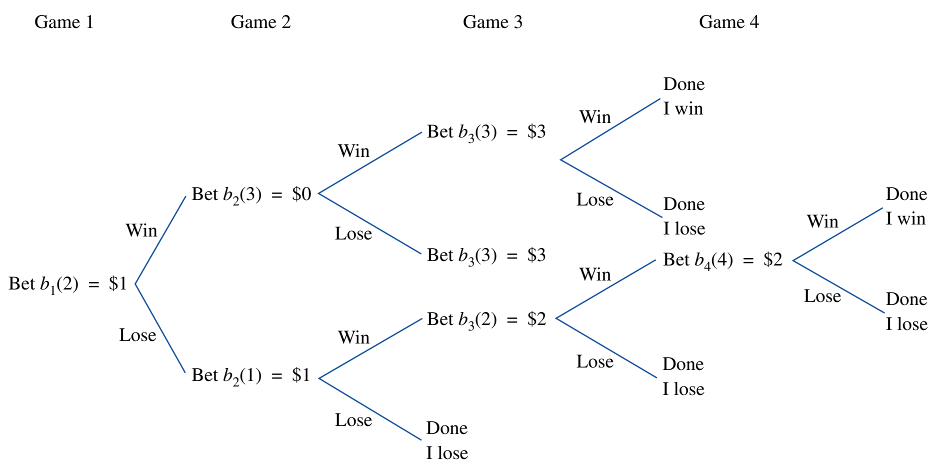


FIGURE 1
Ways Gambler Can Reach \$6

$$f_2(5) = \max \begin{cases} .4f_3(5) + .6f_3(5) = .64^* & \text{(Bet \$0)} \\ .4f_3(6) + .6f_3(4) = .64^* & \text{(Bet \$1)} \\ .4f_3(7) + .6f_3(3) = .64^* & \text{(Bet \$2)} \\ .4f_3(8) + .6f_3(2) = .496 & \text{(Bet \$3)} \\ .4f_3(9) + .6f_3(1) = .40 & \text{(Bet \$4)} \\ .4f_3(10) + .6f_3(0) = .40 & \text{(Bet \$5)} \end{cases}$$

Thus, $f_2(5) = .64$, and $b_2(5) = \$0, \$1,$ or $\$2$. For $d \geq 6$, $f_2(d) = 1$ and $b_2(d) = \$0, \$1, \ldots, \$(d - 6)$.

Stage 1 Computations

$$f_1(2) = \max \begin{cases} .4f_2(2) + .6f_2(2) = .16 & \text{(Bet \$0)} \\ .4f_2(3) + .6f_2(1) = .1984^* & \text{(Bet \$1)} \\ .4f_2(4) + .6f_2(0) = .1984^* & \text{(Bet \$2)} \end{cases}$$

Thus, $f_1(2) = .1984$, and $b_1(2) = \$1$ or $\$2$. Hence, the gambler has a .1984 chance of reaching \$6. Suppose the gambler begins by betting $b_1(2) = \$1$. Then Figure 1 indicates the various possibilities that can occur. By following the strategy outlined in the figure, the gambler can reach her goal of \$6 in two different ways. First, she can win game 1 and game 3. This will occur with probability $(.4)^2 = .16$. Second, the gambler can win if she loses the first game but wins the next three games. This will occur with probability $.6(.4)^3 = .0384$. Hence, the gambler's probability of reaching \$6 is $.16 + .0384 = .1984 = f_1(2)$.

EXAMPLE 4 Tennis Serves

Martina McEnroe has two types of serves: a hard serve (H) and a soft serve (S).[†] The probability that Martina's hard serve will land in bounds is p_H, and the probability that her soft serve will land in bounds is p_S. If Martina's hard serve lands in bounds, there is a probability w_H that Martina will win the point. If Martina's soft serve lands in bounds, there is a probability w_S that Martina will win the point. We assume that $p_H < p_S$ and

[†]Based on material by E. V. Denardo, personal communication.

$w_H > w_S$. Martina's goal is to maximize the probability of winning a point on which she serves. Use dynamic programming to help Martina select an optimal serving strategy. Remember that if both serves are out of bounds, Martina loses the point.

Solution To maximize Martina's probability of winning the point, we give her a reward of 1 if she wins the point and a reward of 0 if she loses the point. We also define $f_t (t = 1, 2)$ to be the probability that Martina wins a point if she plays optimally and is about to take her tth serve. To determine the optimal serving strategy, we work backward, beginning with f_2. If Martina serves hard on the second serve, she will win the point (and earn a reward of 1) with probability $p_H w_H$. Similarly, if she serves soft on the second serve, her expected reward is $p_S w_S$. Thus, we have

$$f_2 = \max \begin{cases} p_H w_H & \text{(Serve hard)} \\ p_S w_S & \text{(Serve soft)} \end{cases}$$

For the moment, let's assume that

$$p_S w_S > p_H w_H \tag{5}$$

If (5) holds, then Martina should serve soft on the second serve. In this situation, $f_2 = p_S w_S$.

To determine f_1, we need to look at what happens on the first serve. If Martina serves hard on the first serve, the events in Table 6 can occur, and Martina earns an expected reward of $p_H w_H + (1 - p_H) f_2$. If Martina serves soft on the first serve, then the events in Table 7 can occur, and Martina's expected reward is $p_S w_S + (1 - p_S) f_2$. We now write the following recursion for f_1:

$$f_1 = \max \begin{cases} p_H w_H + (1 - p_H) f_2 & \text{(Serve hard)} \\ p_S w_S + (1 - p_S) f_2 & \text{(Serve soft)} \end{cases}$$

TABLE 6
Computation of Expected Reward If First Serve Is Hard

Event	Probability of Event	Expected Reward for Given Event
First serve in and Martina wins point	$p_H w_H$	1
First serve in and Martina loses point	$p_H (1 - w_H)$	0
First serve out of bounds	$1 - p_H$	f_2

TABLE 7
Computation of Expected Reward If First Serve Is Soft

Event	Probability of Event	Expected Reward for Given Event
First serve in and Martina wins point	$p_s w_s$	1
First serve in and Martina loses point	$p_s (1 - w_s)$	0
First serve out of bounds	$1 - p_s$	f_2

From this equation, we see that Martina should serve hard on the first serve if

$$p_H w_H + (1 - p_H)f_2 \geq p_S w_S + (1 - p_S)f_2 \tag{6}$$

(If (6) is not satisfied, Martina should serve soft on the first serve.)

Continuing with the assumption that $p_S w_S > w_H p_H$ (which implies that $f_2 = p_S w_S$), we may substitute $f_2 = p_S w_S$ into (6) to obtain the result that Martina should serve hard on the first serve if

$$p_H w_H + (1 - p_H)p_S w_S \geq p_S w_S + (1 - p_S)p_S w_S$$

or

$$p_H w_H \geq p_S w_S(1 + p_H - p_S) \tag{7}$$

For example, if $p_H = .60$, $p_S = .90$, $w_H = .55$, and $w_S = .50$, then (5) and (7) are both satisfied, and Martina should serve hard on her first serve and soft on her second serve. On the other hand, if $p_H = .25$, $p_S = .80$, $w_H = .60$, and $w_S = .45$, then both serves should be soft. The reason for this is that in this case, the hard serve's advantage from the fact that w_H exceeds w_S is outweighed by the fact that a hard serve on the first serve greatly increases the chances of a double fault.

To complete our analysis, we must consider the situation where (5) does not hold. We now show that if

$$p_H w_H \geq p_S w_S \tag{8}$$

Martina should serve hard on both serves. Note that if (8) holds, then $f_2 = \max\{p_H w_H, p_S w_S\} = p_H w_H$, and Martina should serve hard on the second serve. Now (6) implies that Martina should serve hard on the first serve if

$$p_H w_H + (1 - p_H)p_H w_H \geq p_S w_S + (1 - p_S)p_H w_H$$

Upon rearrangement, the last inequality becomes

$$p_H w_H(1 + p_S - p_H) \geq p_S w_S$$

Dividing both sides of the last inequality by $p_S w_S$ shows that Martina should serve hard on the first serve if

$$\frac{p_H w_H}{p_S w_S}(1 + p_S - p_H) \geq 1$$

After noting that $p_H w_H \geq p_S w_S$ and $(1 + p_S - p_H) > 1$ (because $p_S > p_H$), we see that the last inequality holds. Thus, we have shown that if $p_H w_H \geq p_S w_S$, Martina should serve hard on both serves. This is reasonable, because if it is optimal to serve hard on the second (and this requires $p_H w_H \geq p_S w_S$), then it should be optimal to serve hard on the first serve, because the danger of double-faulting (which is the drawback to the hard serve) is less immediate on the first serve. Of course, Example 4 could have been solved using a decision tree; see Problem 10 of Section 13.4.

In our solution to Example 4, we have shown how Martina's optimal strategy depends on the values of the parameters defining the problem. This is a kind of sensitivity analysis like the one applied to linear programming problems in Chapters 5 and 6.

PROBLEMS

Group A

1 Vladimir Ulanowsky is playing Keith Smithson in a two-game chess match. Winning a game scores 1 match point, and drawing a game scores $\frac{1}{2}$ match point. After the two games are played, the player with more match points is declared the champion. If the two players are tied after two games, they continue playing until someone wins a game (the winner of that game will be the champion). During each game, Ulanowsky can play one of two ways: boldly or conservatively. If he plays boldly, he has a 45% chance of winning the game and a 55% chance of losing the game. If he plays conservatively, he has a 90% chance of drawing the game and a 10% chance of losing the game. Ulanowsky's goal is to maximize his probability of winning the match. Use dynamic programming to help him accomplish this goal. If this problem is solved correctly, even though Ulanowsky is the inferior player, his chance of winning the match is over $\frac{1}{2}$. Explain this anomalous result.

2 Dickie Hustler has \$2 and is going to toss an unfair coin (probability .4 of heads) three times. Before each toss, he can bet any amount of money (up to what he now has). If heads comes up, Dickie wins the number of dollars he bets; if tails comes up, he loses the number of dollars he bets. Use dynamic programming to determine a strategy that maximizes Dickie's probability of having at least \$5 after the third coin toss.

Group B

3 Supppose that Army trails by 14 points in the Army–Navy football game. Army's guardian angel has assured the Army coach that his team will have the ball two more times during the game and will score a touchdown (worth 6 points) each time it has the ball. The Army coach has also been assured that Navy will not score any more points. Suppose a win is assigned a value of 1, a tie is .3, and a loss is 0. Army's problem is to determine whether to go for 1 or 2 points after each touchdown. A 1-point conversion is always successful, and a 2-point conversion is successful only 40% of the time. The Army coach wants to maximize the expected reward earned from the outcome of the game. Use dynamic programming to determine an optimal strategy. Then prove the following result: *No matter what value is assigned to a tie, it is never optimal to use the following strategy: Go for a 1-point conversion after the first touchdown and go for a 2-point conversion after the second touchdown.* Note that this (suboptimal) strategy is the one most coaches follow!

19.4 Further Examples of Probabilistic Dynamic Programming Formulations

Many probabilistic dynamic programming problems can be solved using recursions of the following form (for max problems):

$$f_t(i) = \max_a \left\{ (\text{expected reward during stage } t|i, a) + \sum_j p(j|i, a, t) f_{t+1}(j) \right\} \quad \textbf{(9)}$$

In (9), $f_t(i)$ is the maximum expected reward that can be earned during stages $t, t + 1, \ldots$ end of the problem, given that the state at the beginning of stage t is i. The max in (9) is taken over all actions a that are feasible when the state at the beginning of stage t is i. In (9), $p(j|i, a, t)$ is the probability that the next period's state will be j, given that the current (stage t) state is i and action a is chosen. Hence, the summation in (9) represents the expected reward from stage $t + 1$ to the end of the problem. By choosing a to maximize the right-hand side of (9), we are choosing a to maximize the expected reward earned from stage t to the end of the problem, and this is what we want to do. The following are six examples of probabilistic dynamic programming formulations.

EXAMPLE 5 Sunco Oil Drilling

Sunco Oil has D dollars to allocate for drilling at sites $1, 2, \ldots, T$. If x dollars are allocated to site t, the probability is $q_t(x)$ that oil will be found on site t. Sunco estimates that if site t has any oil, it is worth r_t dollars. Formulate a recursion that could be used to enable Sunco to maximize the expected value of all oil found on sites $1, 2, \ldots, T$.

Solution This is a typical resource allocation problem (see Example 1). Therefore, the stage should represent the number of sites, the decision for site t is how many dollars to allocate to site t, and the state is the number of dollars available to allocate to sites $t, t + 1, \ldots, T$. We therefore define $f_t(d)$ to be the maximum expected value of the oil that can be found on sites $t, t + 1, \ldots, T$ if d dollars are available to allocate to sites $t, t + 1, \ldots, T$.

We make the reasonable assumption that $q_T(x)$ is a nondecreasing function of x. If this is the case, then at stage T, all the money should be allocated to site T. This yields

$$f_T(d) = r_T q_T(d) + (1 - q_T(d))0 = r_T q_T(d)$$

For $t < T$,

$$f_t(d) = \max_x \{r_t q_t(x) + f_{t+1}(d - x)\}$$

where x must satisfy $0 \le x \le d$. The last recursion follows, because $r_t q_t(x)$ is the expected value of the reward for stage t, and since Sunco will have $d - x$ dollars available for sites $t + 1, t + 2, \dots, T, f_{t+1}(d - x)$ is the expected value of the oil that can be found by optimally drilling at sites $t + 1, t + 2, \dots, T$. To solve the problem, we would work backward until $f_1(D)$ had been determined.

EXAMPLE 6 Bass Fishing

Each year, the owner of a lake must determine how many bass to capture and sell. During year t, a price p_t will be received for each bass that is caught. If the lake contains b bass at the beginning of year t, the cost of capturing x bass is $c_t(x|b)$. Between the time that year t's bass are caught and year $t + 1$ begins, the bass in the lake multiply by a random factor **D**, where $P(\mathbf{D} = d) = q(d)$.

Formulate a dynamic programming recursion that can be used to determine a bass-catching strategy that will maximize the owner's net profit over the next ten years. At present, the lake contains 10,000 bass.

Solution As in Example 8 of Chapter 18, the stage is the year, the state is the number of bass in the lake at the beginning of the year, and the decision is how many bass to catch during each year. We define $f_t(b)$ to be the maximum expected net profit that can be earned during the years $t, t + 1, \dots, 10$ if the lake contains b bass at the beginning of year t. Then

$$f_{10}(b) = \max_x \{xp_{10} - c_{10}(x|b)\}$$

where $0 \le x \le b$, and for $t < 10$

$$f_t(b) = \max_x \left\{xp_t - c_t(x|b) + \sum_d q(d) f_{t+1}(d(b - x))\right\}$$

In this recursion, x must satisfy $0 \le x \le b$. To justify the recursion for $t < 10$, first note that the profits during year t are (with certainty) $xp_t - c_t(x|b)$. Then with probability $q(d)$, year $t + 1$'s state will be $d(b - x)$. It then follows that if x bass are caught during year t, the maximum expected net profit that can be earned during the years $t + 1, t + 2, \dots,$ 10 will be

$$\sum_d q(d) f_{t+1}(d(b - x))$$

Hence, the recursion chooses the number of bass during year t to maximize the sum of year t profits and future profits. To use this recursion, we work backward until $f_1(10{,}000)$ is computed. Then, after the number of bass in the lake at the beginning of year t is observed, we use the recursion to determine the number of bass that should be caught during year t.

EXAMPLE 7 Waiting in Line

When Sally Mutton arrives at the bank, 30 minutes remain on her lunch break. If Sally makes it to the head of the line and enters service before the end of her lunch break, she earns reward r. However, Sally does not enjoy waiting in lines, so to reflect her dislike for waiting in line, she incurs a cost c for each minute she waits. During a minute in which n people are ahead of Sally, there is a probability $p(x|n)$ that x people will complete their transactions. Suppose that when Sally arrives, 20 people are ahead of her in line. Use dynamic programming to determine a strategy for Sally that will maximize her expected net revenue (reward − waiting costs).

Solution When Sally arrives at the bank, she must decide whether to join the line or to give up and leave. At any later time, she may also decide to leave if it is unlikely that she will be served by the end of her lunch break. If 1 minute remained, Sally's decision would be simple: She should stay in line if and only if her expected reward exceeds the cost of waiting for 1 minute (c). Then we can work backward to a problem with 2 minutes left, and so on. We define $f_t(n)$ to be the maximum expected net reward that Sally can receive from time t to the end of her lunch break if at time t, n people are ahead of her. We let $t = 0$ be the present and $t = 30$ be the end of the problem. Since $t = 29$ is the beginning of the last minute of the problem, we write

$$f_{29}(n) = \max \begin{cases} 0 & \text{(Leave)} \\ rp(n|n) - c & \text{(Stay)} \end{cases}$$

This follows because if Sally chooses to leave at time 29, she earns no reward and incurs no more costs. On the other hand, if she stays at time 29, she will incur a waiting cost of c (a revenue of $-c$) and with probability $p(n|n)$ will enter service and receive a reward r. Thus, if Sally stays, her expected net reward is $rp(n|n) - c$.

For $t < 29$, we write

$$f_t(n) = \max \begin{cases} 0 & \text{(Leave)} \\ rp(n|n) - c + \sum_{k<n} p(k|n) f_{t+1}(n-k) & \text{(Stay)} \end{cases}$$

The last recursion follows, because if Sally stays, she will earn an expected reward (as in the $t = 29$ case) of $rp(n|n) - c$ during the current minute, and with probability $p(k|n)$, there will be $n - k$ people ahead of her; in this case, her expected net reward from time $t + 1$ to time 30 will be $f_{t+1}(n - k)$. If Sally stays, her overall expected reward received from time $t + 1, t + 2, \ldots, 30$ will be

$$\sum_{k<n} p(k|n) f_{t+1}(n-k)$$

Of course, if n people complete their transactions during the current minute, the problem ends, and Sally's future net revenue will be zero.

To determine Sally's optimal waiting policy, we work backward until $f_0(20)$ is computed. If $f_0(20)$ is attained by "stay," Sally stays and sees how many people are ahead of her at time 1. She continues to stay until a situation arises for which the optimal action is "leave" or she begins to be served. In either case, the problem terminates.

Problems in which the decision maker can terminate the problem by choosing a particular action are known as **stopping rule problems;** they often have a special structure that simplifies the determination of optimal policies. See Ross (1983) for more information on stopping rule problems.

EXAMPLE 8 Cash Management Policy

E. J. Korvair Department Store is trying to determine an optimal cash management policy. During each day, the demand for cash may be described by a random variable $\mathbf{D}$, where $p(\mathbf{D} = d) = p(d)$. At the beginning of each day, the store sends an employee to the bank to deposit or withdraw funds. Each bank transaction costs K dollars. Then E. J.'s demand for cash is met by cash left from the previous day plus money withdrawn (or minus money deposited). At the end of the day, the store determines its cash balance at the store. If the cash balance is negative, a shortage cost of s dollars per dollar short is incurred. If the ending balance is positive, a cost of i dollars per dollar held is incurred (because of loss of interest that could have been earned by depositing cash in the bank). At the beginning of day 1, the store has \$10,000 cash on hand and a bank balance of \$100,000. Formulate a dynamic programming model that can be used to minimize the expected cost of filling the store's cash needs for the next 30 days.

Solution To determine how much money should be withdrawn or deposited, E. J. needs to know its cash on hand and bank balance at the beginning of the day. As usual, we let time be the stage. At the beginning of each stage (or day), E. J. must decide how much to withdraw from or deposit in the bank. We let $f_t(c, b)$ be the minimum expected cost incurred by the store during days $t, t + 1, \ldots, 30$, given that at the beginning of day t, the store has c dollars cash at the store and b dollars in the bank.

We observe that

$$f_{30}(c, b) = \min_x \left\{ K\delta(x) + \sum_{d \le c+x} p(d)(c + x - d)i + \sum_{d \ge c+x} p(d)(d - c - x)s \right\} \tag{10}$$

Here, x is the amount of money transferred from the bank to the store (if $x < 0$ money is transferred from the store to the bank). Since the store cannot withdraw more than b dollars from the bank or deposit more than c dollars in the bank, x must satisfy $b \ge x \ge -c$. Also, in (10), $\delta(0) = 0$ and $\delta(x) = 1$ for $x \ne 0$. In short, $K\delta(x)$ picks up the transaction cost (if there is a transaction). If $d \le c + x$, the store will end the day with $c + x - d$ dollars, so a cost of $i(c + x - d)$ is incurred (because of lost interest). Since this occurs with probability $p(d)$, the first sum in (10) represents the expected interest costs incurred during day 30. Also note that if $d \ge c + x$, the store will be $d - c - x$ dollars short, and a shortage cost of $s(d - c - x)$ will be incurred. Again, this cost is incurred with probability $p(d)$. Hence, the second sum in (10) is the expected shortage cost incurred during day 30.

For $t < 30$, we write

$$\begin{aligned} f_t(c, b) = \min_x \Bigg\{ & K\delta(x) + \sum_{d \le c+x} p(d)(c + x - d)i \\ & + \sum_{d \ge c+x} p(d)(d - c - x)s + \sum_{d} p(d) f_{t+1}(c + x - d, b - x) \Bigg\} \end{aligned} \tag{11}$$

As in (10), x must satisfy $b \ge x \ge -c$. Also, the term $K\delta(x)$ and the first two summations yield the expected cost incurred during day t. If day t demand is d, then at the beginning of day $t + 1$, the store will have $c + x - d$ dollars cash on hand and a bank balance of $b - x$. Thus, with probability $p(d)$, the store's expected cost during days $t + 1$, $t + 2, \ldots, 30$ will be $f_{t+1}(c + x - d, b - x)$. Weighting $f_{t+1}(c + x - d, b - x)$ by the probability that day t demand will be d, we see that the last sum in (11) is the expected cost incurred during days $t + 1, t + 2, \ldots, 30$. Hence, (11) is correct. To determine the optimal cash management policy, we would use (10) and (11) to work backward until $f_1(10{,}000, 100{,}000)$ has been computed.

EXAMPLE 9 Parking Spaces

Robert Blue is trying to find a parking place near his favorite restaurant. He is approaching the restaurant from the west, and his goal is to park as nearby as possible. The available parking places are pictured in Figure 2. Robert is nearsighted and cannot see ahead; he can only see whether the space he is at now is empty. When Robert arrives at an empty space, he must decide whether to park there or to continue to look for a closer space. Once he passes a space, he cannot return to it. Robert estimates that the probability that space t is empty is p_t. If he does not end up with a parking space, he is embarrassed and incurs a cost M (M is a big positive number). If he does park in space t, he incurs a cost $|t|$. Show how Robert can use dynamic programming to develop a parking strategy that minimizes

FIGURE 2
Location of Parking Places

$$\to \boxed{-T}\ \boxed{1-T}\ \boxed{2-T}\cdots\boxed{-2}\ \boxed{-1}\ \boxed{0}\ \boxed{1}\ \boxed{2}\cdots\boxed{T}$$

0 = Restaurant

his expected cost.

Solution If Robert is at space T, his problem is easy to solve: park in space T if it is empty; otherwise, incur a cost of M. Then Robert can work backward until he determines what to do at space $-T$. For this reason, we let the space Robert is at represent the stage. In order to make a decision at any stage, all Robert must know is whether or not the space is empty (if a space is not empty, he must continue). Thus, the state at any stage is whether or not the space is empty. Of course, if the space is empty, Robert's decision is whether to take the space or to continue.

We define

$f_t(o)$ = minimum expected cost if Robert is at space t and space t is occupied

$f_t(e)$ = minimum expected cost if Robert is at space t and space t is empty

If Robert is at space T, he will park in the space if it is empty (incurring a cost T) or incur a cost M if the space is occupied. Thus, we have $f_T(o) = M$ and $f_T(e) = T$.

For $t < T$, we write

$$f_t(o) = p_{t+1}f_{t+1}(e) + (1 - p_{t+1})f_{t+1}(o) \tag{12}$$

$$f_t(e) = \min \begin{cases} |t| & \text{(Take space } t) \\ p_{t+1}f_{t+1}(e) + (1 - p_{t+1})f_{t+1}(o) & \text{(Don't take space } t) \end{cases} \tag{13}$$

To justify (12), note that if space t is occupied, Robert must next look at space $t + 1$. With probability p_{t+1}, space $t + 1$ will be empty; in this case, Robert's expected cost will be $f_{t+1}(e)$. Similarly, with probability $(1 - p_{t+1})$, space $t + 1$ will be occupied, and Robert will incur an expected cost of $f_{t+1}(o)$. Thus, Robert's expected cost is

$$p_{t+1}f_{t+1}(e) + (1 - p_{t+1})f_{t+1}(o)$$

To justify (13), note that Robert can either take space t (incurring a cost of $|t|$) or continue. Thus, if Robert continues, his expected cost will be

$$p_{t+1}f_{t+1}(e) + (1 - p_{t+1})f_{t+1}(o)$$

Since Robert wants to minimize his expected cost, (13) follows. By using (12) and (13), Robert can work backward to compute $f_{-T}(e)$ and $f_{-T}(o)$. Then he will continue until he reaches an empty space at some location t for which the minimum in (13) is attained by taking space t. If no such empty space is reached, Robert will not find a space, and he will incur a cost M.

EXAMPLE 10 Safecracker

During month $t (t = 1, 2, \ldots, 60)$, expert safecracker Dirk Stack knows that he will be offered a role in a bank job that will pay him d_t dollars. There is, however, a probability p_t that month t's job will result in his capture. If Dirk is captured, all his money will be lost. Dirk's goal is to maximize his expected asset position at the end of month 60. Formulate a dynamic programming recursion that will help Dirk accomplish his goal. At the beginning of month 1, Dirk has \$50,000.

Solution At the beginning of month 60, Dirk has no future to consider and his problem is easy to solve, so we let time represent the stage. At the beginning of each month, Dirk must decide whether or not to take the current month's job offer. In order to make this decision, Dirk must know how much money he has at the beginning of the month. We define $f_t(d)$ to be Dirk's maximum expected asset position at the end of month 60, given that at the beginning of month t, Dirk has d dollars. Then

$$f_{60}(d) = \max \begin{cases} p_{60}(0) + (1 - p_{60})(d + d_{60}) & \text{(Accept month 60 job)} \\ d & \text{(Reject month 60 job)} \end{cases}$$

This result follows, because if Dirk takes the job during month 60, there is a probability p_{60} that he will be caught and end up with zero dollars and a probability $(1 - p_{60})$ that he will not be caught and end up with $d + d_{60}$ dollars. Of course, if Dirk does not take the month 60 job, he ends month 60 with d dollars.

Extending this reasoning yields, for $t < 60$,

$$f_t(d) = \max \begin{cases} p_t(0) + (1 - p_t)f_{t+1}(d + d_t) & \text{(Accept month } t \text{ job)} \\ f_{t+1}(d) & \text{(Reject month } t \text{ job)} \end{cases}$$

Note that if Dirk accepts month t's job, there is a probability p_t that he will be caught (and end up with zero) and a probability $(1 - p_t)$ that he will successfully complete month t's job and earn d_t dollars. In this case, Dirk will begin month $t + 1$ with $d + d_t$ dollars, and his expected final cash position will be $f_{t+1}(d + d_t)$. Of course, if Dirk rejects the month t job, he begins month $t + 1$ with d dollars, and his expected final cash position will be $f_{t+1}(d)$. Since Dirk wants to maximize his expected cash position at the end of month 60, the recursion follows. By using the recursion, Dirk can work backward to compute $f_1(50{,}000)$. Then he can decide whether to accept the month 1 job. Assuming he has not been caught, he can then determine whether to accept the month 2 job, and so on.

As described in Section 18.8, spreadsheets can be used to solve dynamic programming recursions. See Problems 14 and 15 for some examples of how spreadsheets can be used to solve PDPs.

PROBLEMS

Group A

1 The space shuttle is about to go up on another flight. With probability $p_t(z)$, it will use z type t fuel cells during the flight. The shuttle has room for at most W fuel cells. If at any time during the flight, all the type t fuel cells burn out, a cost c_t will be incurred. Assuming the goal is to minimize the expected cost due to fuel cell shortages, set up a dynamic programming model that could be used to determine how to stock the space shuttle with fuel cells. There are T different types of fuel cells.

2 At the beginning of each year, a firm observes its asset position (call it d) and may invest any amount x ($0 \le x \le d$) in a risky investment. During each year, the money invested doubles with probability p and is completely lost with probability $1 - p$. Independently of this investment, the firm's asset position increases by an amount y with probability q_y (y may be negative). If the firm's asset position is negative at the beginning of a year, it cannot invest any money during that year. The firm initially has \$10,000 in assets and wants to maximize its expected asset position ten years from now. Formulate a dynamic programming recursion that will help accomplish this goal.

3 Consider a machine that may be in any one of the states 0, 1, 2, ... At the beginning of each month, the state of the machine is observed, and it is decided whether to replace or keep the machine. If the machine is replaced, a new state 0 machine arrives instantaneously. It costs R dollars to replace a machine. Each month that a state i machine is in operation, a maintenance cost of $c(i)$ is incurred. If a machine is in state i at the beginning of a month, then with probability p_{ij}, the machine will begin the next month in state j. At the beginning of the first month, we own a state i_0 machine. Assuming that the interest rate is 12% per year, formulate a dynamic programming recursion that could be used to minimize the expected discounted cost incurred during the next T months. Note that if we replace a machine at the beginning of a month, we incur a maintenance cost of $c(0)$ during the month, and with probability p_{0i}, we begin the next month with a state i machine.

4 In the time interval between t and $t - 1$ seconds before the departure of Braneast Airlines Flight 313, there is a probability p_t that the airline will receive a reservation for the flight and a probability $1 - p_t$ that the airline will receive no reservation. The flight can seat up to 100 passengers. At departure time, if r reservations have been accepted by the airline, there is a probability $q(y|r)$ that y passengers will show up for the flight. Each passenger who boards the flight adds \$500 to Braneast's revenues, but each passenger who shows up for the flight and cannot be seated receives \$200 in compensation. Formulate a dynamic programming recursion to enable the airline to maximize its expected revenue from Flight 313. Assume that no reservations are received more than 100,000 seconds before flight time.

5 At the beginning of each week, a machine is either running or broken down. If the machine runs throughout the week, it earns revenues of \$100. If the machine breaks down during a week, it earns no revenue for that week. If the machine is running at the beginning of the week, we may perform maintenance on it to lessen the chance of a breakdown. If the maintenance is performed, a running machine has a .4 chance of breaking down during the week; if maintenance is not performed, a running machine has a .7 chance of breaking down during the week. Maintenance costs \$20 per week. If the machine is broken down at the beginning of the week, it must be replaced or repaired. Both repair and replacement occur instantaneously. Repairing a machine costs \$40, and there is a .4 chance that the repaired machine will break down during the week. Replacing a broken machine costs \$90, but the new machine is guaranteed to run throughout the next week of operation. Use dynamic programming to determine a repair, replacement, and maintenance policy that maximizes the expected net profit earned over a four-week period. Assume that the machine is running at the beginning of the first week.

6 I own a single share of Wivco stock. I must sell my share at the beginning of one of the next 30 days. Each day, the price of the stock changes. With probability $q(x)$, the price tomorrow will increase by x% over today's stock price (x can be negative). For example, with probability $q(5)$, tomorrow's stock price will be 5% higher than today's. Show how dynamic programming can be used to determine a strategy that maximizes the expected revenue earned from selling the share of Wivco stock. Assume that at the beginning of the first day, the stock sells for \$10 per share.

Group B

7 The National Cat Foundling Home encourages people to adopt its cats, but (because of limited funds) it allows each prospective owner to inspect only four cats before choosing one of them to take home. Ten-year-old Sara is eager to adopt a cat and agrees to abide by the following rules. A randomly selected cat is brought for Sara to see, and then Sara must either choose the cat or reject it. If the first cat is rejected, Sara sees another randomly selected cat and must accept or reject it. This procedure continues until Sara has selected her cat. Once Sara rejects a cat, she cannot go back later and choose it as her pet. Determine a strategy for Sara that will maximize her probability of ending up with the cat she actually prefers.

8 Consider the following probabilistic inventory model:

a At the beginning of each period, a firm observes its inventory position.

b Then the firm decides how many units to produce during the current period. It costs $c(x)$ dollars to produce x units during a period.

c With probability $q(d)$, d units are demanded during the period. From units on hand (including the current period's production), the firm satisfies as much of the demand as possible. The firm receives r dollars for each unit sold. For each unit of demand that is unsatisfied, a penalty cost p is incurred. All unsatisfied demand is assumed to be lost. For example, if the firm has 20 units available and current demand is 30, a revenue of $20r$ would be received, and a penalty of $10p$ would be incurred.

d If ending inventory is positive, a holding cost of \$1 per unit is incurred.

e The next period now begins.

The firm's inital inventory is zero, and its goal is to minimize the expected cost over a 100-period horizon. Formulate a dynamic programming recursion that will help the firm accomplish its goal.

9 Martha and Ken Allen want to sell their house. At the beginning of each day, they receive an offer. We assume that from day to day, the sizes of the offers are independent random variables and that the probability that a given day's offer is for j dollars is p_j. An offer may be accepted during the day it is made or at any later date. For each day the house remains unsold, a maintenance cost of c dollars is incurred. The house must be sold within 30 days. Formulate a dynamic programming recursion that Martha and Ken can use to maximize their expected net profit (selling price − maintenance cost). Assume that the maintenance cost for a day is incurred before the current day's offer is received and that each offer is for an integer number of dollars.

10 An advertising firm has D dollars to spend on reaching customers in T separate markets. Market t consists of k_t people. If x dollars are spent on advertising in market t, the probability that a given person in market t will be reached is $p_t(x)$. Each person in market t who is reached will buy c_t units of the product. A person who is not reached will not buy any of the product. Formulate a dynamic programming recursion that could be used to maximize the expected number of units sold in T markets.

11 Georgia Stein is the new owner of the New York Yankees. Each season, Georgia must decide how much money to spend on the free agent draft. During each season, Georgia can spend any amount of money on free agents up to the team's capital position at the beginning of the season. If the Yankees finish in ith place during the season, their capital position increases by $R(i)$ dollars less the amount of money spent in the free agent draft. If the Yankees finished in ith place last season and spend d dollars on free agents during the off-season, the probability that the Yankees will finish in place j during the next season is $p_{ij}(d)(j = 1, 2, \ldots, 7)$. Last season, the Yankees finished in first place, and at the end of the season, they had a capital position of D dollars. Formulate a dynamic programming recursion that will enable the Yankees to maximize their expected cash position at the end of T seasons.

12 Bailey Bliss is the campaign manager for Walter

Glenn's presidential campaign. He has D dollars to allocate to T winner-take-all primaries. If x_t dollars are allocated to primary t, then with probability $p_t(x_t)$, Glenn will win primary t and obtain v_t delegates. With probability $1 - p_t(x_t)$, Glenn loses primary t and obtains no delegates. Glenn needs K delegates to be nominated. Use dynamic programming to help Bliss maximize Glenn's probability of being nominated. What aspect of a real campaign does the present formulation ignore?

13 At 7 A.M., eight people leave their cars for repair at Harry's Auto Repair Shop. If person i's car is ready by time t (7 A.M. = time 0, and so on), he will pay Harry $r_i(t)$ dollars. For example, if person 2's car must be ready by 2 P.M., we may have $r_2(8) = 0$. Harry estimates that with probability $p_i(t)$, it will take t hours to repair person i's car. Formulate a dynamic programming recursion that will enable Harry to maximize his expected revenue for the day. His workday ends at 5 P.M. = time 10.

14 In Example 10, suppose $p_t = t/60$ and $d_t = t$. Using a spreadsheet, solve for Dirk's optimal strategy. (*Hint:* The possible states are 50, 51, . . . , 1,880 (thousands).)

15 In Example 9, assume $T = 10$ and $p_t = |t|/10$. Using a spreadsheet, solve for Robert's optimal strategy.

19.5 Markov Decision Processes[†]

To use dynamic programming in a problem for which the stage is represented by time, one must determine the value of T, the number of time periods over which expected revenue or expected profit is maximized (or expected costs are minimized). T is referred to as the **horizon length.** For instance, in the equipment-replacement problem of Section 18.5, if our goal is to minimize costs over a 30-year period, then $T = 30$. Of course, it may be difficult for a decision maker to determine exactly the most suitable horizon length. In fact, when a decision maker is facing a long horizon and is not sure of the horizon length, it is more convenient to assume that the horizon length is infinite.

Suppose a decision maker's goal is to maximize the expected reward earned over an infinite horizon. In many situations, the expected reward earned over an infinite horizon may be unbounded. For example, if for any state and decision, the reward earned during a period is at least \$3, then the expected reward earned during an infinite number of periods will, no matter what decisions are chosen, be unbounded. In this situation, it is not clear how a decision maker should choose a decision. Two approaches are commonly used to resolve the problem of unbounded expected rewards over an infinite horizon.

1 We can discount rewards (or costs) by assuming that a \$1 reward received during the next period will have the same value as a reward of β dollars ($0 < \beta < 1$) received during the current period. This is equivalent to assuming that the decision maker wants to maximize expected discounted reward. Let M be the maximum reward (over all possible states and choices of decisions) that can be received during a single period. Then the maximum expected discounted reward (measured in terms of current period dollars) that can be received over an infinite period horizon is

$$M + M\beta + M\beta^2 + \cdots = \frac{M}{1 - \beta} < \infty$$

Thus, discounting rewards (or costs) resolves the problem of an infinite expected reward.

2 The decision maker can choose to maximize the expected reward earned per period. Then he or she would choose a decision during each period in an attempt to maximize the average reward per period as given by

$$E\left(\lim_{n\to\infty} \frac{\text{reward earned during periods } 1, 2, \ldots, n}{n}\right)$$

[†]This section covers topics that may be omitted with no loss of continuity.

Thus, if a \$3 reward were earned each period, the total reward earned during an infinite number of periods would be unbounded, but the average reward per period would equal \$3.

In our discussion of infinite horizon problems, we choose to resolve the problem of unbounded expected rewards by discounting rewards by a factor β per period. A brief discussion of the criterion of average reward per period is also included. Infinite horizon probabilistic dynamic programming problems are called **Markov decision processes** (or MDPs).

Description of an MDP

An MDP is described by four types of information:

1 State space

2 Decision set

3 Transition probabilities

4 Expected rewards

State Space

At the beginning of each period, the MDP is in some state i, where i is a member of $S = \{1, 2, \ldots, N\}$. S is referred to as the MDP's **state space.**

Decision Set

For each state i, there is a finite set of allowable decisions, $D(i)$.

Transition Probabilities

Suppose a period begins in state i, and a decision $d \in D(i)$ is chosen. Then with probability $p(j|i, d)$, the next period's state will be j. The next period's state depends only on the current period's state and on the decision chosen during the current period (not on previous states and decisions). This is why we use the term *Markov* decision process.

Expected Rewards

During a period in which the state is i and a decision $d \in D(i)$ is chosen, an expected reward of r_{id} is received.

TABLE 8
Next Period's States of Machines

Present State of Machine	Probability That Machine Begins Next Week As			
	Excellent	Good	Average	Bad
Excellent	.7	.3	—	—
Good	—	.7	.3	—
Average	—	—	.6	.4
Bad	—	—	—	1.0 until replaced

EXAMPLE 11 Machine Replacement

At the beginning of each week, a machine is in one of four conditions (states): excellent (E), good (G), average (A), or bad (B). The weekly revenue earned by a machine in each type of condition is as follows: excellent, \$100; good, \$80; average, \$50; bad, \$10. After observing the condition of a machine at the beginning of the week, we have the option of instantaneously replacing it with an excellent machine, which costs \$200. The quality of a machine deteriorates over time, as shown in Table 8. For this situation, determine the state space, decision sets, transition probabilities, and expected rewards.

Solution The set of possible states is $S = \{E, G, A, B\}$. Let

$$R = \text{replace at beginning of current period}$$
$$NR = \text{do not replace during current period}$$

Since it is absurd to replace an excellent machine, we write

$$D(E) = \{NR\} \qquad D(G) = D(A) = D(B) = \{R, NR\}$$

We are given the following transition probabilities:

$$\begin{array}{llll}
p(E|NR, E) = .7 & p(G|NR, E) = .3 & p(A|NR, E) = 0 & p(B|NR, E) = 0 \\
p(E|NR, G) = 0 & p(G|NR, G) = .7 & p(A|NR, G) = .3 & p(B|NR, G) = 0 \\
p(E|NR, A) = 0 & p(G|NR, A) = 0 & p(A|NR, A) = .6 & p(B|NR, A) = .4 \\
p(E|NR, B) = 0 & p(G|NR, B) = 0 & p(A|NR, B) = 0 & p(B|NR, B) = 1
\end{array}$$

If we replace a machine with an excellent machine, the transition probabilities will be the same as if we had begun the week with an excellent machine. Thus,

$$\begin{aligned}
p(E|G, R) &= p(E|A, R) = p(E|B, R) = .7 \\
p(G|G, R) &= p(G|A, R) = p(G|B, R) = .3 \\
p(A|G, R) &= p(A|A, R) = p(A|B, R) = 0 \\
p(B|G, R) &= p(B|A, R) = p(B|B, R) = 0
\end{aligned}$$

If the machine is not replaced, then during the week, we receive the revenues given in the problem. Therefore, $r_{E,NR} = \$100$, $r_{G,NR} = \$80$, $r_{A,NR} = \$50$, and $r_{B,NR} = \$10$. If we replace a machine with an excellent machine, then no matter what type of machine we had at the beginning of the week, we receive \$100 and pay a cost of \$200. Thus, $r_{E,R} = r_{G,R} = r_{A,R} = r_{B,R} = -\$100$.

In an MDP, what criterion should be used to determine the correct decision? Answering this question requires that we discuss the idea of an **optimal policy** for an MDP.

DEFINITION ■ A **policy** is a rule that specifies how each period's decision is chosen. ■

Period t's decision may depend on the prior history of the process. Thus, period t's decision can depend on the state during periods $1, 2, \ldots, t$ and the decisions chosen during periods $1, 2, \ldots, t - 1$.

DEFINITION ■ A policy δ is a **stationary policy** if whenever the state is i, the policy δ chooses (independently of the period) the same decision (call this decision $\delta(i)$). ■

We let δ represent an arbitrary policy and Δ represent the set of all policies. Then

$\mathbf{X}_t$ = random variable for the state of MDP at the beginning of period t (for example, $\mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_n$)

X_1 = given state of the process at beginning of period 1 (initial state)

d_t = decision chosen during period t

$V_\delta(i)$ = expected discounted reward earned during an infinite number of periods, given that at beginning of period 1, state is i and stationary policy will be δ

Then

$$V_\delta(i) = E_\delta\left(\sum_{t=1}^{t=\infty} \beta^{t-1} r_{\mathbf{X}_t d_t} | X_1 = i\right)$$

where $E_\delta(\beta^{t-1} r_{\mathbf{X}_t d_t} | X_1 = i)$ is the expected discounted reward earned during period t, given that at the beginning of period 1, the state is i and stationary policy δ is followed.

In a maximization problem, we define

$$V(i) = \max_{\delta \in \Delta} V_\delta(i) \tag{14}$$

In a minimization problem, we define

$$V(i) = \min_{\delta \in \Delta} V_\delta(i)$$

DEFINITION ■ If a policy δ^* has the property that for all $i \in S$

$$V(i) = V_{\delta^*}(i)$$

then δ^* is an **optimal policy.** ■

The existence of a single policy δ^* that simultaneously attains all N maxima in (14) is not obvious. If the r_{id}'s are bounded, Blackwell (1962) has shown that an optimal policy exists, and there is always a stationary policy that is optimal. (Even if the r_{id}'s are not bounded, an optimal policy may exist.)

We now consider three methods that can be used to determine an optimal stationary policy:

1 Policy iteration

2 Linear programming

3 Value iteration, or successive approximations

Policy Iteration

Value Determination Equations

Before we can explain the policy iteration method, we need to determine a system of linear equations that can be used to find $V_\delta(i)$ for $i \in S$ and any stationary policy δ. Let $\delta(i)$ be the decision chosen by the stationary policy δ whenever the process begins a period in state i. Then $V_\delta(i)$ can be found by solving the following system of N linear equations, the value determination equations:

$$V_\delta(i) = r_{i,\delta(i)} + \beta \sum_{j=1}^{j=N} p(j|i,\, \delta(i))V_\delta(j) \qquad (i = 1, 2, \ldots, N) \tag{15}$$

To justify (15), suppose we are in state i and we follow a stationary policy δ. The current period is period 1. Then the expected discounted reward earned during an infinite number of periods consists of $r_{i,\delta(i)}$ (the expected reward received during the current period) plus β (expected discounted reward, to beginning of period 2, earned from period 2 onward). But with probability $p(j|i,\delta(i))$, we will begin period 2 in state j and earn an expected discounted reward, back to period 2, of $V_\delta(j)$. Thus, the expected discounted reward, discounted back to the beginning of period 2 and earned from the beginning of period 2 onward, is given by

$$\sum_{j=1}^{j=N} p(j|i,\, \delta(i))V_\delta(j)$$

Equation (15) now follows.

To illustrate the use of the value determination equations, we consider the following stationary policy for the machine replacement example:

$$\delta(E) = \delta(G) = NR \qquad \delta(A) = \delta(B) = R$$

This policy replaces a bad or average machine and does not replace a good or excellent machine. For this policy, (15) yields the following four equations:

$$\begin{aligned} V_\delta(E) &= 100 + .9(.7V_\delta(E) + .3V_\delta(G)) \\ V_\delta(G) &= 80 + .9(.7V_\delta(G) + .3V_\delta(A)) \\ V_\delta(A) &= -100 + .9(.7V_\delta(E) + .3V_\delta(G)) \\ V_\delta(B) &= -100 + .9(.7V_\delta(E) + .3V_\delta(G)) \end{aligned}$$

Solving these equations yields $V_\delta(E) = 687.81$, $V_\delta(G) = 572.19$, $V_\delta(A) = 487.81$, and $V_\delta(B) = 487.81$.

Howard's Policy Iteration Method

We now describe Howard's (1960) policy iteration method for finding an optimal stationary policy for an MDP (max problem).

Step 1 Policy evaluation—Choose a stationary policy δ and use the value determination equations to find $V_\delta(i)(i = 1, 2, \ldots, N)$.

Step 2 Policy improvement—For all states $i = 1, 2, \ldots, N$, compute

$$T_\delta(i) = \max_{d \in D(i)} \left(r_{id} + \beta \sum_{j=1}^{j=N} p(j|i,\, d)V_\delta(j) \right) \tag{16}$$

Since we can choose $d = \delta(i)$ for $i = 1, 2, \ldots, N$, $T_\delta(i) \geq V_\delta(i)$. If $T_\delta(i) = V_\delta(i)$ for $i = 1, 2, \ldots N$, then δ is an optimal policy. If $T_\delta(i) > V_\delta(i)$ for at least one state, then δ is not an optimal policy. In this case, modify δ so that the decision in each state i is the decision attaining the maximum in (16) for $T_\delta(i)$. This yields a new stationary policy δ' for which $V_{\delta'}(i) \geq V_\delta(i)$ for $i = 1, 2, \ldots N$, and for at least one state i', $V_{\delta'}(i') > V_\delta(i')$. Return to step 1, with policy δ' replacing policy δ.

In a minimization problem, we replace max in (16) with min. If $T_\delta(i) = V_\delta(i)$ for $i = 1, 2, \ldots, N$, then δ is an optimal policy. If $T_\delta(i) < V_\delta(i)$ for at least one state, then δ is not an optimal policy. In this case, modify δ so that the decision in each state i is the decision attaining the minimum in (16) for $T_\delta(i)$. This yields a new stationary policy δ' for which $V_{\delta'}(i) \le V_\delta(i)$ for $i = 1, 2, \ldots N$, and for at least one state i', $V_{\delta'}(i') < V_\delta(i')$. Return to step 1, with policy δ' replacing policy δ.

The policy iteration method is guaranteed to find an optimal policy for the machine replacement example after evaluating a finite number of policies. We begin with the following stationary policy:

$$\delta(E) = \delta(G) = NR \qquad \delta(A) = \delta(B) = R$$

For this policy, we have already found that $V_\delta(E) = 687.81$, $V_\delta(G) = 572.19$, $V_\delta(A) = 487.81$, and $V_\delta(B) = 487.81$. We now compute $T_\delta(E)$, $T_\delta(G)$, $T_\delta(A)$, and $T_\delta(B)$. Since NR is the only possible decision in E,

$$T_\delta(E) = V_\delta(E) = 687.81$$

and $T_\delta(E)$ is attained by the decision NR.

$$T_\delta(G) = \max\begin{cases} -100 + .9(.7V_\delta(E) + .3V_\delta(G)) = 487.81 & (R) \\ 80 + .9(.7V_\delta(G) + .3V_\delta(A)) = V_\delta(G) = 572.19^* & (NR) \end{cases}$$

Thus, $T_\delta(G) = 572.19$ is attained by the decision NR.

$$T_\delta(A) = \max\begin{cases} -100 + .9(.7V_\delta(E) + .3V_\delta(G)) = 487.81 & (R) \\ 50 + .9(.6V_\delta(A) + .4V_\delta(B)) = 489.03^* & (NR) \end{cases}$$

Thus, $T_\delta(A) = 489.03$ is attained by the decision NR.

$$T_\delta(B) = \max\begin{cases} -100 + .9(.7V_\delta(E) + .3V_\delta(G)) = V_\delta(B) = 487.81^* & (R) \\ 10 + .9V_\delta(B) = 449.03 & (NR) \end{cases}$$

Thus, $T_\delta(B) = V_\delta(B) = 487.81$. We have found that $T_\delta(E) = V_\delta(E)$, $T_\delta(G) = V_\delta(G)$, $T_\delta(B) = V_\delta(B)$, and $T_\delta(A) > V_\delta(A)$. Thus, the policy δ is not optimal, and the policy δ' given by $\delta'(E) = \delta'(G) = \delta'(A) = NR$, $\delta'(B) = R$, is an improvement over δ. We now return to step 1 and solve the value determination equations for δ'. From (15), the value determination equations for δ' are

$$\begin{aligned} V_{\delta'}(E) &= 100 + .9(.7V_{\delta'}(E) + .3V_{\delta'}(G)) \\ V_{\delta'}(G) &= 80 + .9(.7V_{\delta'}(G) + .3V_{\delta'}(A)) \\ V_{\delta'}(A) &= 50 + .9(.6V_{\delta'}(A) + .4V_{\delta'}(B)) \\ V_{\delta'}(B) &= -100 + .9(.7V_{\delta'}(E) + .3V_{\delta'}(G)) \end{aligned}$$

Solving these equations, we obtain $V_{\delta'}(E) = 690.23$, $V_{\delta'}(G) = 575.50$, $V_{\delta'}(A) = 492.35$, and $V_{\delta'}(B) = 490.23$. Observe that in each state i, $V_{\delta'}(i) > V_\delta(i)$. We now apply the policy iteration procedure to δ'. We compute

$$T_{\delta'}(E) = V_{\delta'}(E) = 690.23$$

$$T_{\delta'}(G) = \max\begin{cases} -100 + .9(.7V_{\delta'}(E) + .3V_{\delta'}(G)) = 490.23 & (R) \\ 80 + .9(.7V_{\delta'}(G) + .3V_{\delta'}(A)) = V_{\delta'}(G) = 575.50^* & (NR) \end{cases}$$

Thus, $T_{\delta'}(G) = V_{\delta'}(G) = 575.50$ is attained by NR.

$$T_{\delta'}(A) = \max\begin{cases} -100 + .9(.7V_{\delta'}(E) + .3V_{\delta'}(G)) = 490.23 & (R) \\ 50 + .9(.6V_{\delta'}(A) + .4V_{\delta'}(B)) = V_{\delta'}(A) = 492.35^* & (NR) \end{cases}$$

Thus, $T_{\delta'}(A) = V_{\delta'}(A) = 492.35$ is attained by NR.

$$T_{\delta'}(B) = \max \begin{cases} -100 + .9(.7V_{\delta'}(E) + .3V_{\delta'}(G)) = V_{\delta}(B) = 490.23^* & (R) \\ 10 + .9V_{\delta'}(B) = 451.21 & (NR) \end{cases}$$

Thus, $T_{\delta'}(B) = V_{\delta'}(B) = 490.23$ is attained by R.

For each state i, $T_{\delta'}(i) = V_{\delta'}(i)$. Thus, δ' is an optimal stationary policy. To maximize expected discounted rewards (profits), a bad machine should be replaced, but an excellent, good, or average machine should not be replaced. If we began period 1 with an excellent machine, an expected discounted reward of \$690.23 could be earned.

Linear Programming

It can be shown (see Ross (1983)) that an optimal stationary policy for a maximization problem can be found by solving the following LP:

$$\min z = V_1 + V_2 + \cdots + V_N$$

$$\text{s.t. } V_i - \beta \sum_{j=1}^{j=N} p(j|i, d)V_j \geq r_{id} \qquad \text{(For each state } i \text{ and each } d \in d(i))$$

All variables urs

For a minimization problem, we solve the following LP:

$$\max z = V_1 + V_2 + \cdots + V_N$$

$$\text{s.t. } V_i - \beta \sum_{j=1}^{j=N} p(j|i, d)V_j \leq r_{id} \qquad \text{(For each state } i \text{ and each } d \in d(i))$$

All variables urs

The optimal solution to these LPs will have $V_i = V(i)$. Also, if a constraint for state i and decision d is binding (has no slack or excess), then decision d is optimal in state i.

REMARKS

1 In the objective function, the coefficient of each V_i may be any positive number.
2 If all the V_i's are nonnegative (this will surely be the case if all the r_{id}'s are nonnegative), we may assume that all variables are nonnegative. If it is possible for some state to have $V(i)$ negative, then we must replace each variable V_i by $V_i' - V_i''$, where both V_i' and V_i'' are nonnegative.
3 With LINDO, we may allow $V(i)$ to be negative with the statement **FREE** *Vi*. With LINGO, use the **@FREE** statement to allow a variable to assume a negative value.

Our machine replacement example yields the following LP:

$$\begin{array}{lll} \min z = V_E + V_G + V_A + V_B & & \\ \text{s.t.} \quad V_E \geq 100 + .9(.7V_E + .3V_G) & & (NR \text{ in } E) \\ \quad V_G \geq 80 + .9(.7V_G + .3V_A) & & (NR \text{ in } G) \\ \quad V_G \geq -100 + .9(.7V_E + .3V_G) & & (R \text{ in } G) \\ \quad V_A \geq 50 + .9(.6V_A + .4V_B) & & (NR \text{ in } A) \\ \quad V_A \geq -100 + .9(.7V_E + .3V_G) & & (R \text{ in } A) \\ \quad V_B \geq 10 + .9V_B & & (NR \text{ in } B) \\ \quad V_B \geq -100 + .9(.7V_E + .3V_G) & & (R \text{ in } B) \\ \quad \text{All variables urs} & & \end{array}$$

The LINDO output for this LP yields $V_E = 690.23$, $V_G = 575.50$, $V_A = 492.35$, and

$V_B = 490.23$. These values agree with those found via the policy iteration method. The LINDO output also indicates that the first, second, fourth, and seventh constraints have no slack. Thus, the optimal policy is to replace a bad machine and not to replace an excellent, good, or average machine.

Value Iteration

There are several versions of value iteration (see Denardo (1982)). We discuss for a maximization problem the simplest value iteration scheme, also known as *successive approximations*. Let $V_t(i)$ be the maximum expected discounted reward that can be earned during t periods if the state at the beginning of the current period is i. Then

$$V_t(i) = \max_{d \in D(i)} \left\{ r_{id} + \beta \sum_{j=1}^{j=N} p(j|i, d) V_{t-1}(j) \right\} \qquad (t \geq 1)$$

$$V_0(i) = 0$$

This result follows, because during the current period, we earn an expected reward (in current dollars) of r_{id}, and during the next $t - 1$ periods, our expected discounted reward (in terms of period 2 dollars) is

$$\sum_{j=1}^{j=N} p(j|i, d) V_{t-1}(j)$$

Let $d_t(i)$ be the decision that must be chosen during period 1 in state i to attain $V_t(i)$. For an MDP with a finite state space and each $D(i)$ containing a finite number of elements, the most basic result in successive approximations states that for $i = 1, 2, \ldots, N$,

$$|V_t(i) - V(i)| \leq \frac{\beta^t}{1 - \beta} \max_{i,d} |r_{id}|$$

Recall that $V(i)$ is the maximum expected discounted reward earned during an infinite number of periods if the state is i at the beginning of the current period. Then

$$\lim_{t \to \infty} d_t(i) = \delta^*(i)$$

where $\delta^*(i)$ defines an optimal stationary policy. Since $\beta < 1$, for t sufficiently large, $V_t(i)$ will come arbitrarily close to $V(i)$. For instance, in the machine replacement example, $\beta = .9$ and $\max |r_{id}| = 100$. Thus, for all states, $V_{50}(i)$ would differ by at most $(.9)^{50}(\frac{100}{.10}) = \$5.15$ from $V(i)$. The equation

$$\lim_{t \to \infty} d_t(i) = \delta^*(i)$$

implies that for t sufficiently large, the decision that is optimal in state i for a t-period problem is also optimal in state i for an infinite horizon problem. This result is reminiscent of the turnpike theorem result for the knapsack problem that was discussed in Chapter 6.

Unfortunately, there is usually no easy way to determine a t^* such that for all i and $t \geq t^*$, $d_t(i) = \delta^*(i)$. (See Denardo (1982) for a partial result in this direction.) Despite this fact, value iteration methods usually obtain a satisfactory approximation to the $V(i)$ and $\delta^*(i)$ with less computational effort than is needed by the policy iteration method or by linear programming. Again, see Denardo (1982) for a discussion of this matter.

We illustrate the computation of V_1 and V_2 for the machine replacement example:

$$V_1(E) = 100 \qquad (NR)$$

$$V_1(G) = \max \begin{cases} 80^* & (NR) \\ -100 & (R) \end{cases} = 80$$

$$V_1(A) = \max \begin{cases} 50^* & (NR) \\ -100 & (R) \end{cases} = 50$$

$$V_1(B) = \max \begin{cases} 10^* & (NR) \\ -100 & (R) \end{cases} = 10$$

The * indicates the action attaining $V_1(i)$. Then

$$V_2(E) = 100 + .9(.7V_1(E) + .3V_1(G)) = 184.6 \qquad (NR)$$

$$V_2(G) = \max \begin{cases} 80 + .9(.7V_1(G) + .3V_1(A)) = 143.9^* & (NR) \\ -100 + .9(.7V_1(E) + .3V_1(G)) = -15.4 & (R) \end{cases}$$

$$V_2(A) = \max \begin{cases} 50 + .9(.6V_1(A) + .4V_1(B)) = 80.6^* & (NR) \\ -100 + .9(.7V_1(E) + .3V_1(G)) = -15.4 & (R) \end{cases}$$

$$V_2(B) = \max \begin{cases} 10 + .9V_1(B) = 19^* & (NR) \\ -100 + .9(.7V_1(E) + .3V_1(G)) = -15.4 & (R) \end{cases}$$

The * now indicates the decision $d_2(i)$ attaining $V_2(i)$. Observe that after two iterations of successive aproximations, we have not yet come close to the actual values of $V(i)$ and have not found it optimal to replace even a bad machine.

In general, if we want to ensure that all the $V_t(i)$'s are within ϵ of the corresponding $V(i)$, we would perform t^* iterations of successive approximations, where

$$\frac{\beta^{t^*}}{1 - \beta} \max_{i,d} |r_{id}| < \epsilon$$

There is no guarantee, however, that after t^* iterations of successive approximations, the optimal stationary policy will have been found.

Maximizing Average Reward per Period

We now briefly discuss how linear programming can be used to find a stationary policy that maximizes the expected per-period reward earned over an infinite horizon. Consider a decision rule or policy Q that chooses decision $d \in D(i)$ with probability $q_i(d)$ during a period in which the state is i. A policy Q will be a stationary policy if each $q_i(d)$ equals 0 or 1. To find a policy that maximizes expected reward per period over an infinite horizon, let π_{id} be the fraction of all periods in which the state is i and the decision $d \in D(i)$ is chosen. Then the expected reward per period may be written as

$$\sum_{i=1}^{i=N} \sum_{d \in D(i)} \pi_{id} r_{id} \tag{17}$$

What constraints must be satisfied by the π_{id}? First, all π_{id}'s must be nonnegative. Second,

$$\sum_{i=1}^{i=N} \sum_{d \in D(i)} \pi_{id} = 1$$

must hold. Finally, the fraction of all periods during which a transition occurs out of state

j must equal the fraction of all periods during which a transition occurs into state j. This is identical to the restriction on steady-state probabilities for Markov chains discussed in Section 17.5. This yields (for $j = 1, 2, \dots, n$),

$$\sum_{d \in D(j)} \pi_{jd}(1 - p(j|j, d)) = \sum_{d \in D(i)} \sum_{i \neq j} \pi_{id} p(j|i, d)$$

Rearranging the last equality yields (for $j = 1, 2, \dots, N$)

$$\sum_{d \in D(j)} \pi_{jd} = \sum_{d \in D(i)} \sum_{i=1}^{i=N} \pi_{id} p(j|i, d)$$

Putting together our objective function (17) and all the constraints yields the following LP:

$$\begin{aligned} \max z &= \sum_{i=1}^{i=N} \sum_{d \in D(i)} \pi_{id} r_{id} \\ \text{s.t.} \quad & \sum_{i=1}^{i=N} \sum_{d \in D(i)} \pi_{id} = 1 \\ & \sum_{d \in D(j)} \pi_{jd} = \sum_{d \in D(i)} \sum_{i=1}^{i=N} \pi_{id} p(j|i, d) \\ & (j = 1, 2, \dots, N) \\ & \text{All } \pi_{id's} \geq 0 \end{aligned} \tag{18}$$

It can be shown that this LP has an optimal solution in which for each i, at most one $\pi_{id} > 0$. This optimal solution implies that expected reward per period is maximized by a solution in which each $q_i(d)$ equals 0 or 1. Thus, the optimal solution to (18) will occur for a stationary policy. For states having $\pi_{id} = 0$, any decision may be chosen without affecting the expected reward per period.

We illustrate the use of (18) for Example 11 (machine replacement). For this example, (18) yields

$$\begin{aligned} \max z &= 100\pi_{ENR} + 80\pi_{GNR} + 50\pi_{ANR} + 10\pi_{BNR} - 100(\pi_{GR} + \pi_{AR} + \pi_{BR}) \\ \text{s.t.} \quad & \pi_{ENR} + \pi_{GNR} + \pi_{ANR} + \pi_{BNR} + \pi_{GR} + \pi_{AR} + \pi_{BR} = 1 \\ & \pi_{ENR} = .7(\pi_{ENR} + \pi_{GR} + \pi_{AR} + \pi_{BR}) \\ & \pi_{GNR} + \pi_{GR} = .3(\pi_{GR} + \pi_{AR} + \pi_{BR} + \pi_{ENR}) + .7\pi_{GNR} \\ & \pi_{AR} + \pi_{ANR} = .3\pi_{GNR} + .6\pi_{ANR} \\ & \pi_{BR} + \pi_{BNR} = \pi_{BNR} + .4\pi_{ANR} \end{aligned}$$

Using LINDO, we find the optimal objective function value for this LP to be $z = 60$. The only nonzero decision variables are $\pi_{ENR} = .35$, $\pi_{GNR} = .50$, $\pi_{AR} = .15$. Thus, an average of \$60 profit per period can be earned by not replacing an excellent or good machine but replacing an average machine. Since we are replacing an average machine, the action chosen during a period in which a machine is in bad condition is of no importance.

PROBLEMS

Group A

1 A warehouse has an end-of-period capacity of 3 units. During a period in which production takes place, a setup cost of \$4 is incurred. A \$1 holding cost is assessed against

TABLE 9

Last Month's Sales	Current Month's Sales	
	Good	Bad
Good	.95	.05
Bad	.40	.60

TABLE 10

Last Month's Sales	Current Month's Sales	
	Good	Bad
Good	.80	.20
Bad	.20	.80

each unit of a period's ending inventory. Also, a variable production cost of \$1 per unit is incurred. During each period, demand is equally likely to be 1 or 2 units. All demand must be met on time, and $\beta = .8$. The goal is to minimize expected discounted costs over an infinite horizon.

a Use the policy iteration method to determine an optimal stationary policy.

b Use linear programming to determine an optimal stationary policy.

c Perform two iterations of value iteration.

2 Priceler Auto Corporation must determine whether or not to give consumers 8% or 11% financing on new cars. If Priceler gives 8% financing during the current month, the probability distribution of sales during the current month will be as shown in Table 9. If Priceler gives 11% financing during the current month, the probability distribution of sales during the current month will be as shown in Table 10. "Good" sales represents 400,000 sales per month, "bad" sales represents 300,000 sales per month. For example, if last month's sales were bad and Priceler gives 8% financing during the current month, there is a .40 chance that sales will be good during the current month. At 11% financing rates, Priceler earns \$1,000 per car, and at 8% financing, Priceler earns \$800 per car. Priceler's goal is to maximize expected discounted profit over an infinite horizon (use $\beta = .98$).

a Use the policy iteration method to determine an optimal stationary policy.

b Use linear programming to determine an optimal stationary policy.

c Perform two iterations of value iteration.

d Find a policy that maximizes average profit per month.

3 Suppose you are using the policy iteration method to determine an optimal policy for an MDP. How might you use LINDO to solve the value determination equations?

Group B

TABLE 11

Today's Price	Tomorrow's Price			
	\$0	\$1	\$2	\$3
\$0	.5	.3	.1	.1
\$1	.1	.5	.2	.2
\$2	.2	.1	.5	.2
\$3	.1	.1	.3	.5

4 During any day, I may own either 0 or 1 share of a stock. The price of the stock is governed by the Markov chain shown in Table 11. At the beginning of a day in which I own a share of stock, I may either sell it at today's price or keep it. At the beginning of a day in which I don't own a share of stock, I may either buy a share of stock at today's price or not buy a share. My goal is to maximize my expected discounted profit over an infinite horizon (use $\beta = .95$).

a Use the policy iteration method to determine an optimal stationary policy.

b Use linear programming to determine an optimal stationary policy.

c Perform two iterations of value iteration.

d Find a policy that maximizes average daily profit.

5 Ethan Sherwood owns two printing presses, on which he prints two types of jobs. At the beginning of each day, there is a .5 probability that a type 1 job will arrive, a .1 probability that a type 2 job will arrive, and a .4 probability that no job will arrive. Ethan receives \$400 for completing a type 1 job and \$200 for completing a type 2 job. (Payment for each job is received in advance.) Each type of job takes an average of three days to complete. To model this, we assume that each day a job is in press there is a $\frac{1}{3}$ probability that its printing will be completed at the end of the day. If both presses are busy at the beginning of the day, any arriving job is lost to the system. The crucial decision is when (if ever) Ethan should accept the less profitable type 2 job. Ethan's goal is to maximize expected discounted profit (use $\beta = .90$).

a Use the policy iteration method to determine an optimal stationary policy.

b Use linear programming to determine an optimal stationary policy.

c Perform two iterations of value iteration.

SUMMARY

Key to Formulating Probabilistic Dynamic Programming Problems (PDPs)

Suppose the possible states during period $t + 1$ are $s_1, s_2, \ldots s_n$, and the probability that the period $t + 1$ state will be s_i is p_i. Then the minimum expected cost incurred during periods $t + 1, t + 2, \ldots,$ end of the problem is

$$\sum_{i=1}^{i=n} p_i f_{t+1}(s_i)$$

where $f_{t+1}(s_i)$ is the minimum expected cost incurred from period $t + 1$ to the end of the problem, given that the state during period $t + 1$ is s_i.

Maximizing the Probability of a Favorable Event Occurring

To maximize the probability that a favorable event will occur, assign a reward of 1 if the favorable event occurs and a reward of 0 if it does not occur.

Markov Decision Processes

A **Markov decision process** (MDP) is simply an infinite-horizon PDP. Let $V_\delta(i)$ be the expected discounted reward earned during an infinite number of periods, given that at the beginning of period 1, the state is i and the stationary policy δ is followed.

For a maximization problem, we define

$$V(i) = \max_{\delta \in \Delta} V_\delta(i)$$

For a minimization problem, we define

$$V(i) = \min_{\delta \in \Delta} V_\delta(i)$$

If a policy δ^* has the property that for all $i \in S$,

$$V(i) = V_{\delta^*}(i)$$

then δ^* is an **optimal policy.** We can use the **value determination equations** to determine $V_\delta(i)$:

$$V_\delta(i) = r_{i,\delta(i)} + \beta \sum_{j=1}^{j=N} p(j|i, \delta(i))V_\delta(j) \qquad (i = 1, 2, \ldots, N) \tag{15}$$

An optimal policy for an MDP may be determined by one of three methods:

1 Policy iteration

2 Linear programming

3 Value iteration, or successive approximations

Policy Iteration

A summary of Howard's policy iteration method for a maximization problem follows.

Step 1 Policy evaluation—Choose a stationary policy δ and use the value determination equations to find $V_\delta(i)(i = 1, 2, \ldots, N)$.

Step 2 Policy improvement—For all states $i = 1, 2, \ldots, N$, compute

$$T_\delta(i) = \max_{d \in D(i)} \left\{ r_{id} + \beta \sum_{j=1}^{j=N} p(j|i, d)V_\delta(j) \right\} \tag{16}$$

Since we can choose $d = \delta(i)$ for $i = 1, 2, \ldots, N$, $T_\delta(i) \geq V_\delta(i)$. If $T_\delta(i) = V_\delta(i)$ for $i = 1, 2, \ldots, N$, then δ is an optimal policy. If $T_\delta(i) > V_\delta(i)$ for at least one state, then δ is not an optimal policy. In this case, modify δ so that the decision in each state i is the decision attaining the maximum in (16) for $T_\delta(i)$. This yields a new stationary policy δ' for which $V_{\delta'}(i) \geq V_\delta(i)$ for $i = 1, 2, \ldots, N$, and for at least one state i', $V_{\delta'}(i') > V_\delta(i')$. Return to step 1, with policy δ' replacing policy δ.

Linear Programming

In a maximization problem, $V(i)$ for each state may be determined by solving the following LP:

$$\min z = V_1 + V_2 + \cdots + V_N$$

$$\text{s.t.} \quad V_i - \beta \sum_{j=1}^{j=N} p(j|i, d)V_j \geq r_{id} \qquad \text{(For each state } i \text{ and each } d \in D(i))$$

All variables urs

If the constraint for state i and decision d has no slack, then decision d is optimal in state i.

Value Iteration, or Successive Approximations

Let $V_t(i)$ be the maximum expected discounted reward that can be earned during t periods if the state at the beginning of the current period is i. Then

$$V_t(i) = \max_{d \in D(i)} \left\{ r_{id} + \beta \sum_{j=1}^{j=N} p(j|i, d)V_{t-1}(j) \right\} \qquad (t \geq 1)$$

$$V_0(i) = 0$$

As t grows large, $V_t(i)$ will approach $V(i)$. For t sufficiently large, the decision that is optimal in state i for a t-period problem is also optimal in state i for an infinite-horizon problem.

TABLE 12

No. of Sales Reps Assigned to District	Sales (millions)		
	District 1	District 2	District 3
0	\$1, \$4	\$2, \$5	\$3, \$4
1	\$2, \$6	\$4, \$6	\$5, \$5
2	\$3, \$7	\$5, \$6	\$6, \$7
3	\$4, \$8	\$6, \$6	\$7, \$7

REVIEW PROBLEMS

Group A

1 A company has five sales representatives available for assignment to three sales districts. The sales in each district during the current year depend on the number of sales representatives assigned to the district and on whether the national economy has a bad or good year (see Table 12). In the Sales column for each district, the first number represents sales if the national economy had a bad year, and the second number represents sales if the economy had a good year. There is a .3 chance that the national economy will have a good year and a .7 chance that the national economy will have a bad year. Use dynamic programming to determine an assignment of sales representatives to districts that maximizes the company's expected sales.

2 At the beginning of each period, a company must determine how many units to produce. A setup cost of \$5 is incurred during each period in which production takes place. The production of each unit also incurs a \$2 variable cost. All demand must be met on time, and there is a \$1 per-unit holding cost on each period's ending inventory. During each period, it is equally likely that demand will equal 0 or 1 unit. Assume that each period's ending inventory cannot exceed 2 units.

a Use dynamic programming to minimize the expected costs incurred during three periods. Assume that the initial inventory is 0 units.

b Now suppose that each unit demanded can be sold for \$4. If the demand is not met on time, the sale is lost. Use dynamic programming to maximize the expected profit earned during three periods. Assume that the initial inventory is 0 units.

c In parts (a) and (b), is an (s, S) policy optimal?

3 At Hot Dog Queen Restaurant, the following sequence of events occurs during each minute:

a With probability p, a customer arrives and waits in line.

b Hot Dog Queen determines the rate s at which customers are served. If any customers are in the restaurant, then with probability s, one of the customers completes service and leaves the restaurant. It costs $c(s)$ dollars per period to serve customers at a rate s. Each customer spends R dollars, and the customer's food costs Hot Dog Queen $R - 1$ dollars to prepare.

c For each customer in line at the end of the minute, a cost of h dollars is assessed (because of customer inconvenience).

d The next minute begins.

Formulate a recursion that could be used to maximize expected revenues less costs (including customer inconvenience costs) incurred during the next T minutes. Assume that initially there are no customers present.

4 At the beginning of 2004, the United States has B barrels of oil. If x barrels of oil are consumed during a year, then consumers earn a benefit (measured in dollars) of $u(x)$. The United States may spend money on oil exploration. If d dollars are spent during a year on oil exploration, then there is a probability $p(d)$ that an oil field (containing 500,000 barrels of oil) will be found. Formulate a recursion that can be used to maximize the expected discounted benefits less exploration expenditures earned from the beginning of 2004 to the end of the year 2539.

5 I am a contestant on the popular TV show "Tired of Fortune." During the bonus round, I will be asked up to four questions. For each question that is correctly answered, I win a certain amount of money. One incorrect answer, however, means that I lose all the money I have previously won, and the game is over. If I elect to pass, or not answer a question, the game is over, but I may keep what I have already won. The amount of money I win for each correct question and the probability that I will answer each question correctly are shown in Table 13.

a My goal is to maximize the expected amount of money won. Use dynamic programming to accomplish this goal.

b Suppose that I am allowed to pass, or not answer a question, and still go on to the next question. Now determine how to maximize the amount of money won.

TABLE 13

Question	Probability of Correct Answer	Money Won
1	.6	\$10,000
2	.5	\$20,000
3	.4	\$30,000
4	.3	\$40,000

6 A machine in excellent condition earns \$100 profit per week, a machine in good condition earns \$70 per week, and a machine in bad condition earns \$20 per week. At the beginning of any week, a machine may be sent out for repairs at a cost of \$90. A machine that is sent out for repairs returns in excellent condition at the beginning of the next week. If a machine is not repaired, the condition of the machine evolves in accordance with the Markov chain shown in Table 14. The company wants to maximize its expected discounted profit over an infinite horizon ($\beta = .9$).

a Use policy iteration to determine an optimal stationary policy.

b Use linear programming to detemine an optimal stationary policy.

c Perform two iterations of value iteration.

TABLE 14

	Next Week		
This Week	Excellent	Good	Bad
Excellent	.7	.2	.1
Good	0	.7	.3
Bad	0	.1	.9

7 A country now has 10 units of capital. Each year, it may consume any amount of the available capital and invest the rest. Invested capital has a 50% chance of doubling and a 50% chance of losing half its value. For example, if the country invests 6 units of capital, there is a 50% chance that the 6 units will turn into 12 capital units and a 50% chance that the invested capital will turn into 3 units. What strategy should be used to maximize total expected consumption over a four-year period?

8 The Dallas Mavericks trail by two points and have the ball with 10 seconds remaining. They must decide whether to take a two- or a three-point shot. Assume that once the Mavericks take their shot, time expires. The probability that a two-point shot is successful is TWO, and the probability that a three-point shot is successful is THREE. If the game is tied, an overtime period will be played. Assume that there is a .5 chance the Mavericks will win in overtime. (*Note:* This problem is often used on Microsoft job interviews.)

a Give a rule based on the values of TWO and THREE that tells Dallas what to do.

b Typical values for an NBA team are TWO $= .45$ and THREE $= .35$. Based on this information, what strategy should most NBA teams follow?

9 At any time, the size of a tree is 0, 1, 2, or 3. We must decide when to harvest the tree. Each year, it costs \$1 to maintain the tree. It costs \$5 to harvest a tree. The sales price for a tree of each size is as follows:

Tree Size	Sales Price
0	\$20
1	\$30
2	\$45
3	\$49

The transition probability matrix for the size of the tree is as follows:

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & .8 & .2 & 0 & 0 \\ 1 & 0 & .9 & .1 & 0 \\ 2 & 0 & 0 & .7 & .3 \\ 3 & 0 & 0 & 0 & 1 \end{array}$$

For example, 80% of all size 0 trees begin the next year as size 0 trees, and 20% of all size 0 trees begin the next year as size 1 trees. Assuming the discount factor for cash flows is .9 per year, determine an optimal harvesting strategy.

10 For \$50, we can enter a raffle. We draw a certificate containing a number 100, 200, 300, . . . , 1,000. Each number is equally likely. At any time, we can redeem the highest-numbered certificate we have obtained so far for the face value of the certificate. We may enter the raffle as many times as we wish. Assuming no discounting, what strategy would maximize our expected profit? How does this model relate to the problem faced by an unemployed person who is searching for a job?

11 At the beginning of each year, an aircraft engine is in good, fair, or poor condition. It costs \$500,000 to run a good engine for a year, \$1 million to run a fair engine for a year, and \$2 million to run a poor engine for a year. A fair engine can be overhauled for \$2 million, and it immediately becomes a good engine. A poor engine can be replaced for \$3 million, and it immediately becomes a good engine. The transition probability matrix for an engine is as follows:

$$\begin{array}{c|ccc} & \text{Good} & \text{Fair} & \text{Poor} \\ \hline \text{Good} & .7 & .2 & .1 \\ \text{Fair} & 0 & .6 & .4 \\ \text{Poor} & 0 & 0 & 1 \end{array}$$

The discount factor for costs is .9. What strategy minimizes expected discounted cost over an infinite horizon?

Group B

12 A syndicate of college students spends weekends gambling in Las Vegas. They begin week 1 with W dollars. At the beginning of each week, they may wager any amount of their money at the gambling tables. If they wager d dollars, then with probability p, their wealth increases by d dollars, and with probability $1 - p$, their wealth decreases by d dollars. Their goal is to maximize their expected wealth at the end of T weeks.

a Show that if $p \geq \frac{1}{2}$, the students should bet all their money.

b Show that if $p < \frac{1}{2}$, the students should bet no money. (*Hint:* Define $f_t(w)$ as the maximum expected wealth at the end of week T, given that wealth is w dollars at the beginning of week t; by working backward, find an expression for $f_t(w)$.)

Group C

13 You have invented a new product: the HAL DVD player. Each of 1,000 potential customers places a different value on this product. A consumer's valuation is equally likely to be any number between \$0 and \$1,000. It costs \$100 to produce the HAL player. During a year in which we set a price p for the product, all customers valuing the product at \$$p$ or more will purchase the product. Each year, we set a price for the product. What pricing strategy will maximize our expected profit over three years? What commonly observed phenomenon does this problem illustrate?

REFERENCES

The following books contain elementary discussions of Markov decision processes and probabilistic dynamic programming: