CHAPTER 21

Dynamic Programming

CONTENTS

Dynamic programming is an approach to problem solving that decomposes a large problem that may be difficult to solve into a number of smaller problems that are usually much easier to solve. Moreover, the dynamic programming approach allows us to break up a large problem in such a way that once all the smaller problems have been solved, we have an optimal solution to the large problem. We shall see that each of the smaller problems is identified with a stage of the dynamic programming solution procedure. As a consequence, the technique has been applied to decision problems that are multistage in nature. Often, multiple stages are created because a sequence of decisions must be made over time. For example, a problem of determining an optimal decision over a one-year horizon might be broken into 12 smaller stages, where each stage requires an optimal decision over a one-month horizon. In most cases, each of these smaller problems cannot be considered to be completely independent of the others, and it is here that dynamic programming is helpful. Let us begin by showing how to solve a shortest-route problem using dynamic programming.

21.1 A SHORTEST-ROUTE PROBLEM

Let us illustrate the dynamic programming approach by using it to solve a shortest-route problem. Consider the network presented in Figure 21.1. Assuming that the numbers above each arc denote the direct distance in miles between two nodes, find the shortest route from node 1 to node 10.

Before attempting to solve this problem, let us consider an important characteristic of all shortest-route problems. This characteristic is a restatement of Richard Bellman's famous **principle of optimality** as it applies to the shortest-route problem.[1]

Principle of Optimality

If a particular node is on the optimal route, then the shortest path from that node to the end is also on the optimal route.

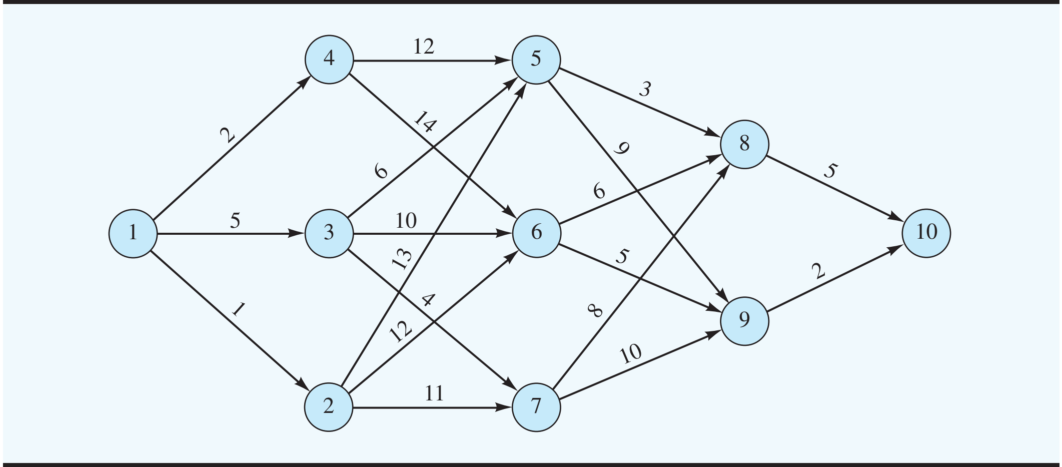
The dynamic programming approach to the shortest-route problem essentially involves treating each node as if it were on the optimal route and making calculations accordingly. In doing so, we will work backward by starting at the terminal node, node 10, and calculating the shortest route from each node to node 10 until we reach the origin, node 1. At this point, we will have solved the original problem of finding the shortest route from node 1 to node 10.

As we stated in the introduction to this chapter, dynamic programming decomposes the original problem into a number of smaller problems that are much easier to solve. In the shortest-route problem for the network in Figure 21.1, the smaller problems that we will create define a four-stage dynamic programming problem. The first stage begins with nodes that are exactly one arc away from the destination, and ends at the destination node. Note from Figure 21.1 that only nodes 8 and 9 are exactly one arc away from node 10. In dynamic programming terminology, nodes 8 and 9 are considered to be the input nodes for stage 1, and node 10 is considered to be the output node for stage 1.

The second stage begins with all nodes that are exactly two arcs away from the destination and ends with all nodes that are exactly one arc away. Hence, nodes 5, 6, and 7 are the input nodes for stage 2, and nodes 8 and 9 are the output nodes for stage 2. Note that the output nodes

[1]S. Dreyfus, *Dynamic Programming and the Calculus of Variations* (New York: Academic Press, 1965).

FIGURE 21.1 NETWORK FOR THE SHORTEST-ROUTE PROBLEM



for stage 2 are the input nodes for stage 1. The input nodes for the third-stage problem are all nodes that are exactly three arcs away from the destination—that is, nodes 2, 3, and 4. The output nodes for stage 3, all of which are one arc closer to the destination, are nodes 5, 6, and 7. Finally, the input node for stage 4 is node 1, and the output nodes are 2, 3, and 4. The decision problem we shall want to solve at each stage is, Which arc is best to travel over in moving from each particular input node to an output node? Let us consider the stage 1 problem.

We arbitrarily begin the stage 1 calculations with node 9. Because only one way affords travel from node 9 to node 10, this route is obviously shortest and requires us to travel a distance of 2 miles. Similarly, only one path goes from node 8 to node 10. The shortest route from node 8 to the end is thus the length of that route, or 5 miles. The stage 1 decision problem is solved. For each input node, we have identified an optimal decision—that is, the best arc to travel over to reach the output node. The stage 1 results are summarized here:

Stage 1

Input Node	**Arc (decision)**	**Shortest Distance to Node 10**
8	8–10	5
9	9–10	2

To begin the solution to the stage 2 problem, we move to node 7. (We could have selected node 5 or 6; the order of the nodes selected at any stage is arbitrary.) Two arcs leave node 7 and are connected to input nodes for stage 1: arc 7–8, which has a length of 8 miles, and arc 7–9, which has a length of 10 miles. If we select arc 7–8, we will have a distance from node 7 to node 10 of 13 miles, that is, the length of arc 7–8, 8 miles, plus the shortest distance to node 10 from node 8, 5 miles. Thus, the decision to select arc 7–8 has a total associated distance of $8 + 5 = 13$ miles. With a distance of 10 miles for arc 7–9 and stage 1 results showing a distance of 2 miles from node 9 to node 10, the decision to select arc 7–9 has an associated distance of $10 + 2 = 12$ miles. Thus, given we are at node 7, we should select arc 7–9 because it is on the path that will reach node 10 in the shortest distance

(12 miles). By performing similar calculations for nodes 5 and 6, we can generate the following stage 2 results:

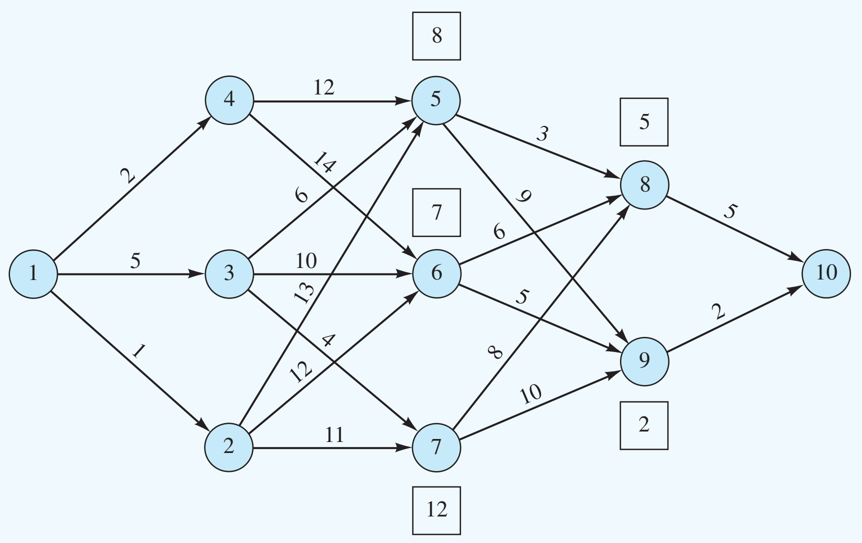
Stage 2

Input Node	Arc (decision)	Output Node	Shortest Distance to Node 10
5	5–8	8	8
6	6–9	9	7
7	7–9	9	12

In Figure 21.2 the number in the square above each node considered so far indicates the length of the shortest route from that node to the end. We have completed the solution to the first two subproblems (stages 1 and 2). We now know the shortest route from nodes 5, 6, 7, 8, and 9 to node 10.

To begin the third stage, let us start with node 2. Note that three arcs connect node 2 to the stage 2 input nodes. Thus, to find the shortest route from node 2 to node 10, we must make three calculations. If we select arc 2–7 and then follow the shortest route to the end, we will have a distance of 11 + 12 = 23 miles. Similarly, selecting arc 2–6 requires 12 + 7 = 19 miles, and selecting arc 2–5 requires 13 + 8 = 21 miles. Thus, the shortest route from node 2 to node 10 is 19 miles, which indicates that arc 2–6 is the best decision, given that we are at node 2. Similarly, we find that the shortest route from node 3 to node 10 is given by Min {4 + 12, 10 + 7, 6 + 8} = 14; the shortest route from node 4 to node 10 is given by Min {14 + 7, 12 + 8} = 20. We complete the stage 3 calculations with the following results:

FIGURE 21.2 INTERMEDIATE SOLUTION TO THE SHORTEST-ROUTE PROBLEM USING DYNAMIC PROGRAMMING



Stage 3

Input Node	Arc (decision)	Output Node	Shortest Distance to Node 10
2	2–6	6	19
3	3–5	5	14
4	4–5	5	20

In solving the stage 4 subproblem, we find that the shortest route from node 1 to node 10 is given by Min {1 + 19, 5 + 14, 2 + 20} = 19. Thus, the optimal decision at stage 4 is the selection of arc 1–3. By moving through the network from stage 4 to stage 3 to stage 2 to stage 1, we can identify the best decision at each stage and therefore the shortest route from node 1 to node 10.

Stage	Arc (decision)
4	1–3
3	3–5
2	5–8
1	8–10

Thus, the shortest route is through nodes 1–3–5–8–10 with a distance of 5 + 6 + 3 + 5 = 19 miles.

Note how the calculations at each successive stage make use of the calculations at prior stages. This characteristic is an important part of the dynamic programming procedure. Figure 21.3 illustrates the final network calculations. Note that in working back through the stages we have now determined the shortest route from every node to node 10.

Dynamic programming, while enumerating or evaluating several paths at each stage, does not require us to enumerate all possible paths from node 1 to node 10. Returning to the stage 4 calculations, we consider three alternatives for leaving node 1. The complete route associated with each of these alternatives is presented as follows:

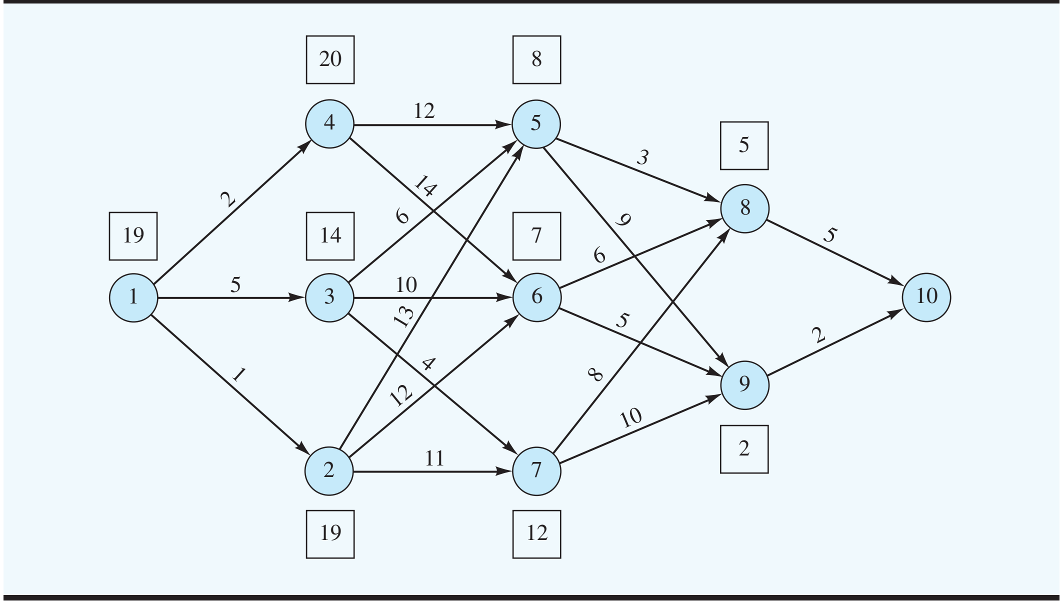
Arc Alternatives at Node 1	Complete Path to Node 10	Distance	
1–2	1–2–6–9–10	20	
1–3	1–3–5–8–10	19	← Selected as best
1–4	1–4–5–8–10	22	

Try Problem 2, part (a), for practice solving a shortest-route problem using dynamic programming.

When you realize that there are a total of 16 alternate routes from node 1 to node 10, you can see that dynamic programming has provided substantial computational savings over a total enumeration of all possible solutions.

The fact that we did not have to evaluate all the paths at each stage as we moved backward from node 10 to node 1 is illustrative of the power of dynamic programming. Using dynamic programming, we need only make a small fraction of the number of calculations

FIGURE 21.3 FINAL SOLUTION TO THE SHORTEST-ROUTE PROBLEM USING DYNAMIC PROGRAMMING



that would be required using total enumeration. If the example network had been larger, the computational savings provided by dynamic programming would have been even greater.

21.2 DYNAMIC PROGRAMMING NOTATION

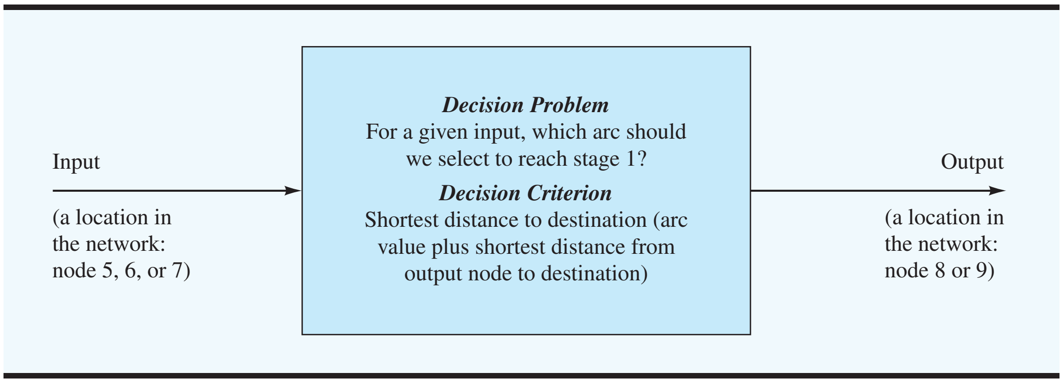
Perhaps one of the most difficult aspects of learning to apply dynamic programming involves understanding the notation. The notation we will use is the same as that used by Nemhauser[2] and is fairly standard.

The **stages** of a dynamic programming solution procedure are formed by decomposing the original problem into a number of subproblems. Associated with each subproblem is a stage in the dynamic programming solution procedure. For example, the shortest-route problem introduced in the preceding section was solved using a four-stage dynamic programming solution procedure. We had four stages because we decomposed the original problem into the following four subproblems:

1. **Stage 1 Problem:** Where should we go from nodes 8 and 9 so that we will reach node 10 along the shortest route?
2. **Stage 2 Problem:** Using the results of stage 1, where should we go from nodes 5, 6, and 7 so that we will reach node 10 along the shortest route?
3. **Stage 3 Problem:** Using the results of stage 2, where should we go from nodes 2, 3, and 4 so that we will reach node 10 along the shortest route?
4. **Stage 4 Problem:** Using the results of stage 3, where should we go from node 1 so that we will reach node 10 along the shortest route?

[2]G. L. Nemhauser, *Introduction to Dynamic Programming* (New York: Wiley, 1966).

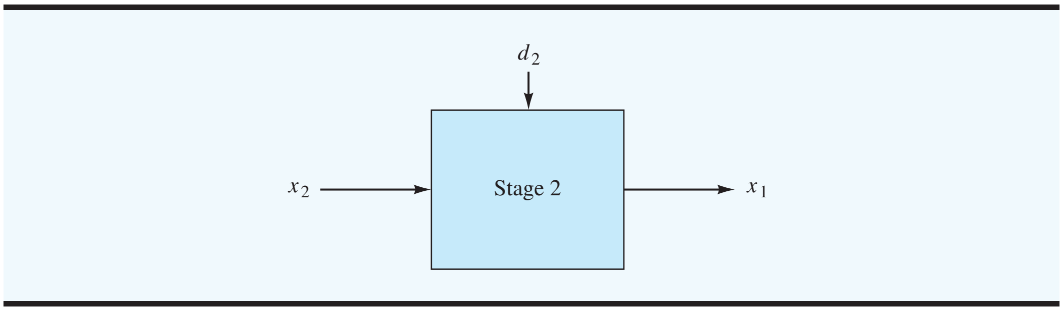
Let us look closely at what occurs at the stage 2 problem. Consider the following representation of this stage:


Input
(a location in the network: node 5, 6, or 7)
Decision Problem
For a given input, which arc should we select to reach stage 1?
Decision Criterion
Shortest distance to destination (arc value plus shortest distance from output node to destination)
Output
(a location in the network: node 8 or 9)

Using dynamic programming notation, we define

x_2 = input to stage 2; represents the location in the network at the beginning of stage 2 (node 5, 6, or 7)
d_2 = decision variable at stage 2 (the arc selected to move to stage 1)
x_1 = output for stage 2; represents the location in the network at the end of stage 2 (node 8 or 9)

Using this notation, the stage 2 problem can be represented as follows:

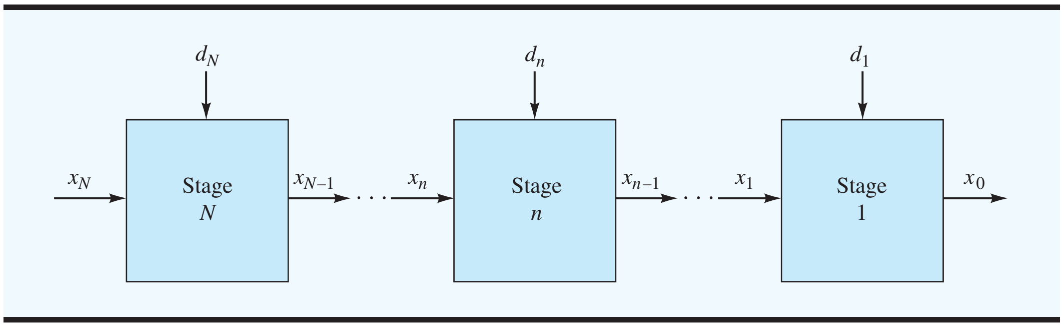


Recall that using dynamic programming to solve the shortest-route problem, we worked backward through the stages, beginning at node 10. When we reached stage 2, we did not know x_2 because the stage 3 problem had not yet been solved. The approach used was to consider *all* alternatives for the input x_2. Then we determined the best decision d_2 for each of the inputs x_2. Later, when we moved forward through the system to recover the optimal sequence of decisions, we saw that the stage 3 decision provided a specific x_2, node 5, and from our previous analysis we knew the best decision (d_2) to make as we continued on to stage 1.

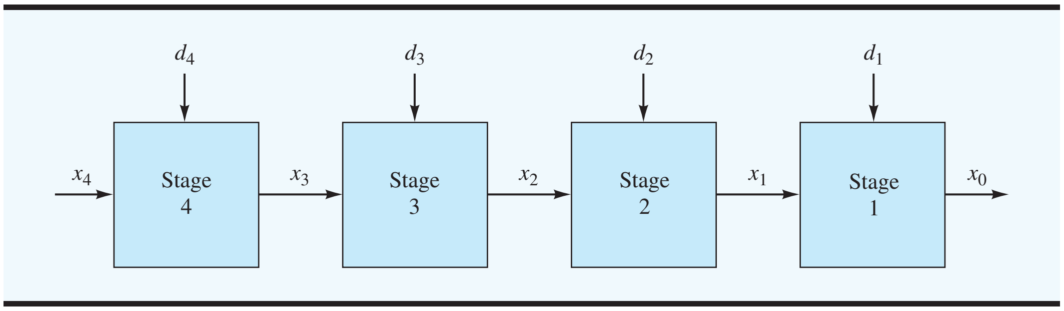
Let us consider a general dynamic programming problem with N stages and adopt the following general notation:

x_n = input to stage n (output from stage $n + 1$)
d_n = **decision variable** at stage n

The general N-stage problem is decomposed as follows:



The four-stage shortest-route problem can be represented as follows:



The values of the input and output variables x_4, x_3, x_2, x_1, and x_0 are important because they join the four subproblems together. At any stage, we will ultimately need to know the input x_n to make the best decision d_n. These x_n variables can be thought of as defining the *state* or condition of the system as we move from stage to stage. Accordingly, these variables are referred to as the **state variables** of the problem. In the shortest-route problem, the state variables represented the location in the network at each stage (i.e., a particular node).

At stage 2 of the shortest-route problem, we considered the input x_2 and made the decision d_2 that would provide the shortest distance to the destination. The output x_1 was based on a combination of the input and the decision; that is, x_1 was a function of x_2 and d_2. In dynamic programming notation, we write:

$$x_1 = t_2(x_2, d_2)$$

where $t_2(x_2, d_2)$ is the function that determines the stage 2 output. Because $t_2(x_2, d_2)$ is the function that "transforms" the input to the stage into the output, this function is referred to as the **stage transformation function.** The general expression for the stage transformation function is

$$x_{n-1} = t_n(x_n, d_n)$$

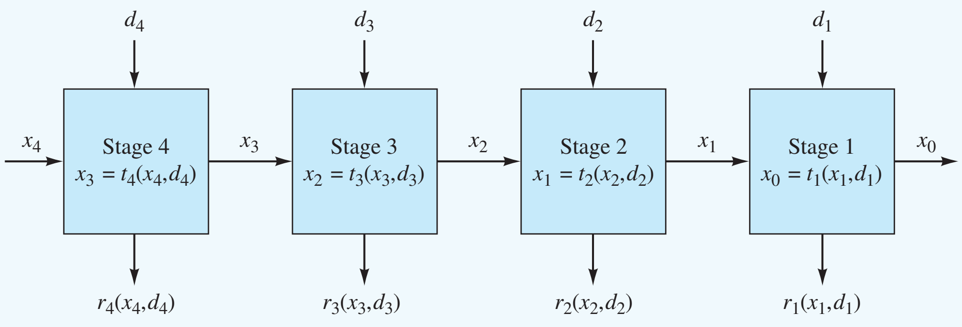
The mathematical form of the stage transformation function is dependent on the particular dynamic programming problem. In the shortest-route problem, the transformation function was based on a tabular calculation. For example, Table 21.1 shows the stage transformation function $t_2(x_2, d_2)$ for stage 2. The possible values of d_2 are the arcs selected in the body of the table.

TABLE 21.1 STAGE TRANSFORMATION $x_1 = t_2(x_2, d_2)$ FOR STAGE 2 WITH THE VALUE OF x_1 CORRESPONDING TO EACH VALUE OF x_2

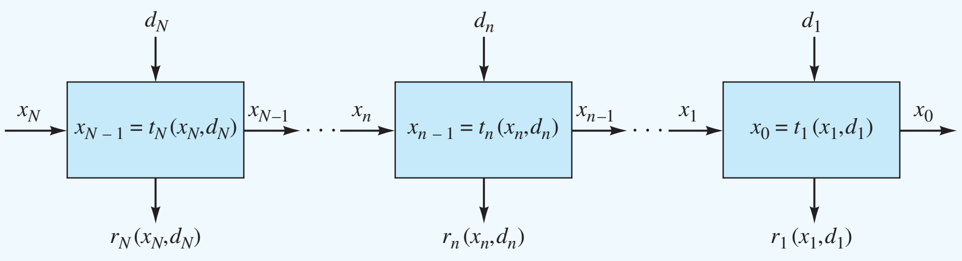
x_2 Input State	x_1 Output State 8	x_1 Output State 9
5	5–8	5–9
6	6–8	6–9
7	7–8	7–9

Each stage also has a return associated with it. In the shortest-route problem, the return was the arc distance traveled in moving from an input node to an output node. For example, if node 7 were the input state for stage 2 and we selected arc 7–9 as d_2, the return for that stage would be the arc length, 10 miles. The return at a stage, which may be thought of as the payoff or value for a stage, is represented by the general notation $r_n(x_n, d_n)$.

Using the stage transformation function and the **return function,** the shortest-route problem can be shown as follows.



If we view a system or a process as consisting of N stages, we can represent a dynamic programming formulation as follows:



Each of the rectangles in the diagram represents a stage in the process. As indicated, each stage has two inputs: the state variable and the decision variable. Each stage also has two outputs: a new value for the state variable and a return for the stage. The new value for the state variable is determined as a function of the inputs using $t_n(x_n, d_n)$. The value of the return for a stage is also determined as a function of the inputs using $r_n(x_n, d_n)$.

The optimal total return depends only on the state variable.

In addition, we will use the notation $f_n(x_n)$ to represent the optimal total return from stage n and all remaining stages, given an input of x_n to stage n. For example, in the shortest-route problem, $f_2(x_2)$ represents the optimal total return (i.e., the minimum distance) from stage 2 and all remaining stages, given an input of x_2 to stage 2. Thus, we see from Figure 21.3 that $f_2(x_2 = \text{node } 5) = 8$, $f_2(x_2 = \text{node } 6) = 7$, and $f_2(x_2 = \text{node } 7) = 12$. These values are the ones indicated in the squares at nodes 5, 6, and 7.

NOTES AND COMMENTS

1. The primary advantage of dynamic programming is its "divide and conquer" solution strategy. Using dynamic programming, a large, complex problem can be divided into a sequence of smaller interrelated problems. By solving the smaller problems sequentially, the optimal solution to the larger problem is found. Dynamic programming is a general approach to problem solving; it is not a specific technique such as linear programming, which can be applied in the same fashion to a variety of problems. Although some characteristics are common to all dynamic programming problems, each application requires some degree of creativity, insight, and expertise to recognize how the larger problems can be broken into a sequence of interrelated smaller problems.
2. Dynamic programming has been applied to a wide variety of problems including inventory control, production scheduling, capital budgeting, resource allocation, equipment replacement, and maintenance. In many of these applications, periods such as days, weeks, and months provide the sequence of interrelated stages for the larger multiperiod problem.

21.3 THE KNAPSACK PROBLEM

The basic idea of the **knapsack problem** is that N different types of items can be put into a knapsack. Each item has a certain weight associated with it as well as a value. The problem is to determine how many units of each item to place in the knapsack to maximize the total value. A constraint is placed on the maximum weight permissible.

To provide a practical application of the knapsack problem, consider a manager of a manufacturing operation who must make a biweekly selection of jobs from each of four categories to process during the following two-week period. A list showing the number of jobs waiting to be processed is presented in Table 21.2. The estimated time required for completion and the value rating associated with each job are also shown.

The value rating assigned to each job category is a subjective score assigned by the manager. A scale from 1 to 20 is used to measure the value of each job, where 1 represents jobs of the least value, and 20 represents jobs of most value. The value of a job depends on such things as expected profit, length of time the job has been waiting to be processed, priority, and so on. In this situation, we would like to select certain jobs during the next two weeks such that all the jobs selected can be processed within 10 working days and the total value of the jobs selected is maximized. In knapsack problem terminology, we are in essence selecting the best jobs for the two-week (10 working days) knapsack, where the knapsack has a capacity equal to the 10-day production capacity. Let us formulate and solve this problem using dynamic programming.

TABLE 21.2 JOB DATA FOR THE MANUFACTURING OPERATION

Job Category	Number of Jobs to Be Processed	Estimated Completion Time per Job (days)	Value Rating per Job
1	4	1	2
2	3	3	8
3	2	4	11
4	2	7	20

This problem can be formulated as a dynamic programming problem involving four stages. At stage 1, we must decide how many jobs from category 1 to process; at stage 2, we must decide how many jobs from category 2 to process; and so on. Thus, we let

d_n = number of jobs processed from category n (decision variable at stage n)
x_n = number of days of processing time remaining at the beginning of stage n (state variable for stage n)

Thus, with a two-week production period, $x_4 = 10$ represents the total number of days available for processing jobs. The stage transformation functions are as follows:

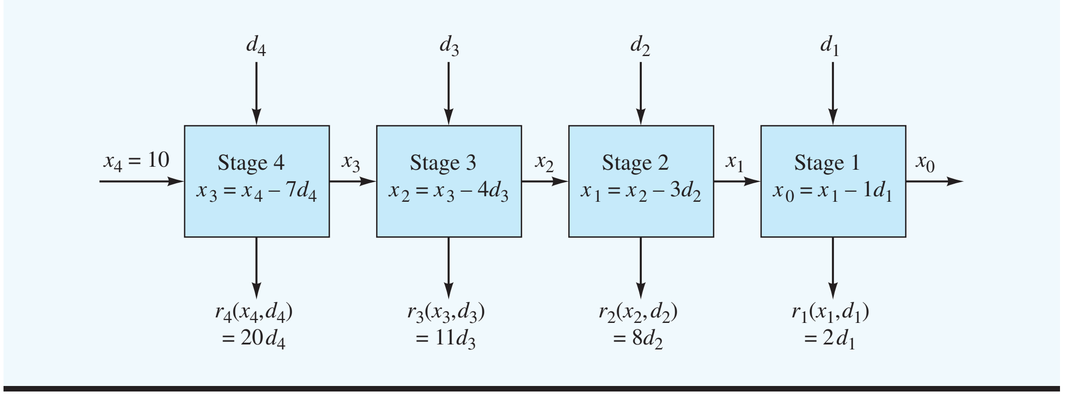
Stage 4. $x_3 = t_4(x_4, d_4) = x_4 - 7d_4$
Stage 3. $x_2 = t_3(x_3, d_3) = x_3 - 4d_3$
Stage 2. $x_1 = t_2(x_2, d_2) = x_2 - 3d_2$
Stage 1. $x_0 = t_1(x_1, d_1) = x_1 - 1d_1$

The return at each stage is based on the value rating of the associated job category and the number of jobs selected from that category. The return functions are as follows:

Stage 4. $r_4(x_4, d_4) = 20d_4$
Stage 3. $r_3(x_3, d_3) = 11d_3$
Stage 2. $r_2(x_2, d_2) = 8d_2$
Stage 1. $r_1(x_1, d_1) = 2d_1$

Figure 21.4 shows a schematic of the problem.

FIGURE 21.4 DYNAMIC PROGRAMMING FORMULATION OF THE JOB SELECTION PROBLEM



As with the shortest-route problem in Section 21.1, we will apply a backward solution procedure; that is, we will begin by assuming that decisions have already been made for stages 4, 3, and 2 and that the final decision remains (how many jobs from category 1 to select at stage 1). A restatement of the principle of optimality can be made in terms of this problem. That is, regardless of whatever decisions have been made at previous stages, if the decision at stage n is to be part of an optimal overall strategy, the decision made at stage n must necessarily be optimal for all remaining stages.

Let us set up a table that will help us calculate the optimal decisions for stage 1.

Stage 1. Note that stage 1's input (x_1), the number of days of processing time available at stage 1, is unknown because we have not yet identified the decisions at the previous stages. Therefore, in our analysis at stage 1, we will have to consider all possible values of x_1 and identify the best decision d_1 for each case; $f_1(x_1)$ will be the total return after decision d_1 is made. The possible values of x_1 and the associated d_1 and $f_1(x_1)$ values are as follows:

x_1	d_1^*	$f_1(x_1)$
0	0	0
1	1	2
2	2	4
3	3	6
4	4	8
5	4	8
6	4	8
7	4	8
8	4	8
9	4	8
10	4	8

The d_1^* column gives the optimal values of d_1 corresponding to a particular value of x_1, where x_1 can range from 0 to 10. The specific value of x_1 will depend on how much processing time has been used by the jobs in the other categories selected in stages 2, 3, and 4. Because each stage 1 job requires one day of processing time and has a positive return of two per job, we always select as many jobs at this stage as possible. The number of category 1 jobs selected will depend on the processing time available, but cannot exceed four.

Recall that $f_1(x_1)$ represents the value of the optimal total return from stage 1 and all remaining stages, given an input of x_1 to stage 1. Therefore, $f_1(x_1) = 2x_1$ for values of $x_1 \leq 4$, and $f_1(x_1) = 8$ for values of $x_1 > 4$. The optimization of stage 1 is accomplished. We now move on to stage 2 and carry out the optimization at that stage.

Stage 2. Again, we will use a table to help identify the optimal decision. Because stage 2's input (x_2) is unknown, we have to consider all possible values from 0 to 10. Also, we have to consider all possible values of d_2 (i.e., 0, 1, 2, or 3). The entries under the heading $r_2(x_2, d_2) + f_1(x_1)$ represent the total return that will be forthcoming from the final two stages, given the input of x_2 and the decision of d_2. For example, if stage 2 were entered with $x_2 = 7$ days of processing time remaining, and if a decision were made to select two jobs from category 2 (i.e., $d_2 = 2$), the total return for stages 1 and 2 would be 18.

d_2	$r_2(x_2,d_2) + f_1(x_1)$						$x_1 = t_2(x_2,d_2^*)$
x_2	0	1	2	3	d_2^*	$f_2(x_2)$	$= x_2 - 3d_2^*$
0	⓪	—	—	—	0	0	0
1	②	—	—	—	0	2	1
2	④	—	—	—	0	4	2
3	6	⑧	—	—	1	8	0
4	8	⑩	—	—	1	10	1
5	8	⑫	—	—	1	12	2
6	8	14	⑯	—	2	16	0
7	8	16	⑱	—	2	18	1
8	8	16	⑳	—	2	20	2
9	8	16	22	㉔	3	24	0
10	8	16	24	㉖	3	26	1

The return for stage 2 would be $r_2(x_2, d_2) = 8d_2 = 8(2) = 16$, and with $x_2 = 7$ and $d_2 = 2$, we would have $x_1 = x_2 - 3d_2 = 7 - 6 = 1$. From the previous table, we see that the optimal return from stage 1 with $x_1 = 1$ is $f_1(1) = 2$. Thus, the total return corresponding to $x_2 = 7$ and $d_2 = 2$ is given by $r_2(7,2) + f_1(1) = 16 + 2 = 18$. Similarly, with $x_2 = 5$, and $d_2 = 1$, we get $r_2(5,1) + f_1(2) = 8 + 4 = 12$. Note that some combinations of x_2 and d_2 are not feasible. For example, with $x_2 = 2$ days, $d_2 = 1$ is infeasible because category 2 jobs each require 3 days to process. The infeasible solutions are indicated by a dash.

After all the total returns in the rectangle have been calculated, we can determine an optimal decision at this stage for each possible value of the input or state variable x_2. For example, if $x_2 = 9$, we can select one of four possible values for d_2: 0, 1, 2, or 3. Clearly $d_2 = 3$ with a value of 24 yields the maximum total return for the last two stages. Therefore, we record this value in the column. For additional emphasis, we circle the element inside the rectangle corresponding to the optimal return. The optimal total return, given that we are in state $x_2 = 9$ and must pass through two more stages, is thus 24, and we record this value in the $f_2(x_2)$ column. Given that we enter stage 2 with $x_2 = 9$ and make the optimal decision there of $d_2^* = 3$, we will enter stage 1 with $x_1 = t_2(9, 3) = x_2 - 3d_2 = 9 - 3(3) = 0$. This value is recorded in the last column in the table. We can now go on to stage 3.

Stage 3. The table we construct here is much the same as for stage 2. The entries under the heading $r_3(x_3, d_3) + f_2(x_2)$ represent the total return over stages 3, 2, and 1 for all possible inputs x_3 and all possible decisions d_3.

d_3 / x_3	$r_3(x_3, d_3) + f_2(x_2)$					$x_2 = t_3(x_3, d_3^*)$
	0	1	2	d_3^*	$f_3(x_3)$	$= x_3 - 4d_3^*$
0	0	—	—	0	0	0
1	2	—	—	0	2	1
2	4	—	—	0	4	2
3	8	—	—	0	8	3
4	10	11	—	1	11	0
5	12	13	—	1	13	1
6	16	15	—	0	16	6
7	18	19	—	1	19	3
8	20	21	22	2	22	0
9	24	23	24	0,2	24	9,1
10	26	27	26	1	27	6

Some features of interest appear in this table that were not present at stage 2. We note that if the state variable $x_3 = 9$, then two possible decisions will lead to an optimal total return from stages 1, 2, and 3; that is, we may elect to process no jobs from category 3, in which case, we will obtain no return from stage 3, but will enter stage 2 with $x_2 = 9$. Because $f_2(9) = 24$, the selection of $d_3 = 0$ would result in a total return of 24. However, a selection of $d_3 = 2$ also leads to a total return of 24. We obtain a return of $11(d_3) = 11(2) = 22$ for stage 3 and a return of 2 for the remaining two stages because $x_2 = 1$. To show the available alternative optimal solutions at this stage, we have placed two entries in the d_3^* and $x_2 = t_3(x_3, d_3^*)$ columns. The other entries in this table are calculated in the same manner as at stage 2. Let us now move on to the last stage.

Stage 4. We know that 10 days are available in the planning period; therefore, the input to stage 4 is $x_4 = 10$. Thus, we have to consider only one row in the table, corresponding to stage 4.

d_4 / x_4	$r_4(x_4, d_4) + f_3(x_3)$				$x_3 = t_4(x_4, d_4^*)$
	0	1	d_4^*	$f_4(x_4)$	$= 10 - 7d_4^*$
10	27	28	1	28	3

The optimal decision, given $x_4 = 10$, is $d_4^* = 1$.

We have completed the dynamic programming solution of this problem. To identify the overall optimal solution, we must now trace back through the tables, beginning at stage 4, the last stage considered. The optimal decision at stage 4 is $d_4^* = 1$. Thus, $x_3 = 10 - 7d_4^* = 3$, and we enter stage 3 with 3 days available for processing. With $x_3 = 3$, we see that the best decision at stage 3 is $d_3^* = 0$. Thus, we enter stage 2 with $x_2 = 3$. The optimal decision at stage 2 with $x_2 = 3$ is $d_2^* = 1$, resulting in $x_1 = 0$. Finally, the decision at stage 1 must be $d_1^* = 0$. The optimal strategy for the manufacturing operation is as follows:

Decision		**Return**
$d_1^* = 0$		0
$d_2^* = 1$		8
$d_3^* = 0$		0
$d_4^* = 1$		20
	Total	28

We should schedule one job from category 2 and one job from category 4 for processing over the next 10 days.

Another advantage of the dynamic programming approach can now be illustrated. Suppose we wanted to schedule the jobs to be processed over an eight-day period only. We can solve this new problem simply by making a recalculation at stage 4. The new stage 4 table would appear as follows:

d_4 / x_4	$r_4(x_4, d_4) + f_3(x_3)$				$x_3 = t_4(x_4, d_4^*)$
	0	1	d_4^*	$f_4(x_4)$	$= 8 - 7d_4^*$
8	22	22	0,1	22	8,1

Actually, we are testing the sensitivity of the optimal solution to a change in the total number of days available for processing. We have here the case of alternative optimal solutions. One solution can be found by setting $d_4^* = 0$ and tracing through the tables. Doing so, we obtain the following:

Decision		**Return**
$d_1^* = 0$		0
$d_2^* = 0$		0
$d_3^* = 2$		22
$d_4^* = 0$		0
	Total	22

A second optimal solution can be found by setting $d_4^* = 1$ and tracing back through the tables. Doing so, we obtain another solution (which has exactly the same total return):

Decision		**Return**
$d_1^* = 1$		2
$d_2^* = 0$		0
$d_3^* = 0$		0
$d_4^* = 1$		20
	Total	22

Can you now solve a knapsack problem using dynamic programming? Try Problem 3.

From the shortest-route and the knapsack examples you should be familiar with the stage-by-stage solution procedure of dynamic programming. In the next section we show how dynamic programming can be used to solve a production and inventory control problem.

21.4 A PRODUCTION AND INVENTORY CONTROL PROBLEM

Suppose we developed forecasts of the demand for a particular product over several periods, and we would like to decide on a production quantity for each of the periods so that demand can be satisfied at a minimum cost. Two costs need to be considered: production costs and holding costs. We will assume that one production setup will be made each period; thus, setup costs will be constant. As a result, setup costs are not considered in the analysis.

We allow the production and holding costs to vary across periods. This provision makes the model more flexible because it also allows for the possibility of using different facilities for production and storage in different periods. Production and storage capacity constraints, which may vary across periods, will be included in the model. We adopt the following notation:

N = number of periods (stages in the dynamic programming formulation)
D_n = demand during stage n; $n = 1, 2, \dots, N$
x_n = a state variable representing the amount of inventory on hand at the beginning of stage n; $n = 1, 2, \dots, N$
d_n = production quantity for stage n; $n = 1, 2, \dots, N$
P_n = production capacity in stage n; $n = 1, 2, \dots, N$
W_n = storage capacity at the end of stage n; $n = 1, 2, \dots, N$
C_n = production cost per unit in stage n; $n = 1, 2, \dots, N$
H_n = holding cost per unit of ending inventory for stage n; $n = 1, 2, \dots, N$

We develop the dynamic programming solution for a problem covering three months of operation. The data for the problem are presented in Table 21.3. We can think of each month as a stage in a dynamic programming formulation. Figure 21.5 shows a schematic of such a formulation. Note that the beginning inventory in January is one unit.

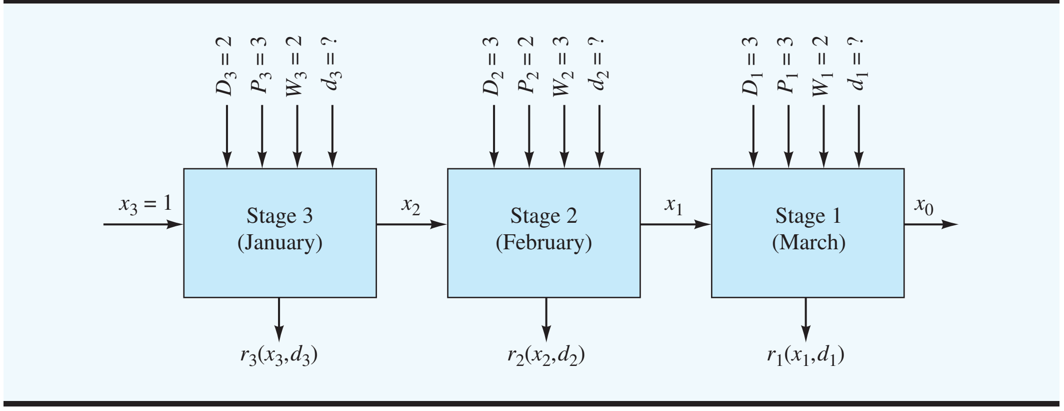
In Figure 21.5 we numbered the periods backward; that is, stage 1 corresponds to March, stage 2 corresponds to February, and stage 3 corresponds to January. The stage transformation

TABLE 21.3 PRODUCTION AND INVENTORY CONTROL PROBLEM DATA

		Capacity		Cost per Unit	
Month	**Demand**	**Production**	**Storage**	**Production**	**Holding**
January	2	3	2	$175	$30
February	3	2	3	150	30
March	3	3	2	200	40

The beginning inventory for January is one unit.

FIGURE 21.5 PRODUCTION AND INVENTORY CONTROL PROBLEM AS A THREE-STAGE DYNAMIC PROGRAMMING PROBLEM



functions take the form of ending inventory = beginning inventory + production − demand. Thus, we have

$$\begin{aligned} x_3 &= 1 \\ x_2 &= x_3 + d_3 - D_3 = x_3 + d_3 - 2 \\ x_1 &= x_2 + d_2 - D_2 = x_2 + d_2 - 3 \\ x_0 &= x_1 + d_1 - D_1 = x_1 + d_1 - 3 \end{aligned}$$

The return functions for each stage represent the sum of production and holding costs for the month. For example, in stage 1 (March), $r_1(x_1, d_1) = 200d_1 + 40(x_1 + d_1 - 3)$ represents the total production and holding costs for the period. The production costs are \$200 per unit, and the holding costs are \$40 per unit of ending inventory. The other return functions are

$$\begin{aligned} r_2(x_2, d_2) &= 150d_2 + 30(x_2 + d_2 - 3) \qquad \text{Stage 2, February} \\ r_3(x_3, d_3) &= 175d_3 + 30(x_3 + d_3 - 2) \qquad \text{Stage 3, January} \end{aligned}$$

This problem is particularly interesting because three constraints must be satisfied at each stage as we perform the optimization procedure. The first constraint is that the ending inventory must be less than or equal to the warehouse capacity. Mathematically, we have

$$x_n + d_n - D_n \leq W_n$$

or

$$x_n + d_n \leq W_n + D_n \qquad (21.1)$$

The second constraint is that the production level in each period may not exceed the production capacity. Mathematically, we have

$$d_n \leq P_n \qquad (21.2)$$

In order to satisfy demand, the third constraint is that the beginning inventory plus production must be greater than or equal to demand. Mathematically, this constraint can be written as

$$x_n + d_n \geq D_n \tag{21.3}$$

Let us now begin the stagewise solution procedure. At each stage, we want to minimize $r_n(x_n, d_n) + f_{n-1}(x_{n-1})$ subject to the constraints given by equations (21.1), (21.2), and (21.3).

Stage 1. The stage 1 problem is as follows:

$$\begin{aligned}
&\text{Min} \quad r_1(x_1, d_1) = 200d_1 + 40(x_1 + d_1 - 3)\\
&\text{s.t.}\\
&\qquad x_1 + d_1 \leq 5 \quad \text{Warehouse constraint}\\
&\qquad\qquad d_1 \leq 3 \quad \text{Production constraint}\\
&\qquad x_1 + d_1 \geq 3 \quad \text{Satisfy demand constraint}
\end{aligned}$$

Combining terms in the objective function, we can rewrite the problem:

$$\begin{aligned}
&\text{Min} \quad r_1(x_1, d_1) = 240d_1 + 40x_1 - 120\\
&\text{s.t.}\\
&\qquad x_1 + d_1 \leq 5\\
&\qquad\qquad d_1 \leq 3\\
&\qquad x_1 + d_1 \geq 3
\end{aligned}$$

Following the tabular approach we adopted in Section 21.3, we will consider all possible inputs to stage 1 (x_1) and make the corresponding minimum-cost decision. Because we are attempting to minimize cost, we will want the decision variable d_1 to be as small as possible and still satisfy the demand constraint. Thus, the table for stage 1 is as follows:

		$f_1(x_1) = r_1(x_1, d_1^*)$
x_1	d_1^*	$240d_1 + 40x_1 - 120$
0	3	600
1	2	400
2	1	200
3	0	0

Production capacity of 3 for stage 1 limits d_1

Warehouse capacity of 3 from stage 2 limits value of x_1

Demand constraint: $x_1 + d_1 \geq 3$

Now let us proceed to stage 2.

Stage 2.

$$\begin{aligned}\text{Min}\quad r_2(x_2, d_2) + f_1(x_1) &= 150d_2 + 30(x_2 + d_2 - 3) + f_1(x_1)\\ &= 180d_2 + 30x_2 - 90 + f_1(x_1)\end{aligned}$$

s.t.

$$\begin{aligned} x_2 + d_2 &\le 6\\ d_2 &\le 2\\ x_2 + d_2 &\ge 3\end{aligned}$$

The stage 2 calculations are summarized in the following table:

d_2 / x_2	$r_2(x_2, d_2) + f_1(x_1)$					
	0	1	2	d_2^*	$f_2(x_2)$	$x_1 = x_2 + d_2^* - 3$
0	—	—	—	—	M	—
1	—	—	900	2	900	0
2	—	750	730	2	730	1

Production capacity of 2 for stage 2

Warehouse capacity of 2 from stage 3

Check demand constraint $x_2 + d_2 \ge 3$ for each x_2, d_2 combination (— indicates an infeasible solution)

The detailed calculations for $r_2(x_2, d_2) + f_1(x_1)$ when $x_2 = 1$ and $d_2 = 2$ are as follows:

$$r_2(1,2) + f_1(0) = 180(2) + 30(1) - 90 + 600 = 900$$

For $r_2(x_2, d_2) + f_1(x_1)$ when $x_2 = 2$ and $d_2 = 1$, we have

$$r_2(2,1) + f_1(0) = 180(1) + 30(2) - 90 + 600 = 750$$

For $x_2 = 2$ and $d_2 = 2$, we have

$$r_2(2,2) + f_1(1) = 180(2) + 30(2) - 90 + 400 = 730$$

Note that an arbitrarily high cost M is assigned to the $f_2(x_2)$ column for $x_2 = 0$. Because an input of 0 to stage 2 does not provide a feasible solution, the M cost associated with the $x_2 = 0$ input will prevent $x_2 = 0$ from occurring in the optimal solution.

Stage 3.

$$\begin{aligned}\text{Min}\quad r_3(x_3, d_3) + f_2(x_2) &= 175d_3 + 30(x_3 + d_3 - 2) + f_2(x_2)\\ &= 205d_3 + 30x_3 - 60 + f_2(x_2)\end{aligned}$$

s.t.

$$\begin{aligned} x_3 + d_3 &\le 4\\ d_3 &\le 3\\ x_3 + d_3 &\ge 2\end{aligned}$$

TABLE 21.4 OPTIMAL PRODUCTION AND INVENTORY CONTROL POLICY

Month	Beginning Inventory	Production	Production Cost	Ending Inventory	Holding Cost	Total Monthly Cost
January	1	2	$ 350	1	$30	$ 380
February	1	2	300	0	0	300
March	0	3	600	0	0	600
		Totals	$1250		$30	$1280

With $x_3 = 1$ already defined by the beginning inventory level, the table for stage 3 becomes

Production capacity of 3 at stage 3

d_3	$r_3(x_3, d_3) + f_2(x_2)$						
x_3	0	1	2	3	d_3^*	$f_3(x_3)$	$x_2 = x_3 + d_3^* - 2$
1	—	M	$\boxed{1280}$	1315	2	1280	1

Try Problem 10 for practice using dynamic programming to solve a production and inventory control problem.

Thus, we find that the total cost associated with the optimal production and inventory policy is $1280. To find the optimal decisions and inventory levels for each period, we trace back through each stage and identify x_n and d_n^* as we go. Table 21.4 summarizes the optimal production and inventory policy.

NOTES AND COMMENTS

1. With dynamic programming, as with other management science techniques, the computer can be a valuable computational aid. However, because dynamic programming is a general approach with stage decision problems differing substantially from application to application, no one algorithm or computer software package is available for solving dynamic programs. Some software packages exist for specific types of problems; however, most new applications of dynamic programming will require specially designed software if a computer solution is to be obtained.
2. The introductory illustrations of dynamic programming presented in this chapter are deterministic and involve a finite number of decision alternatives and a finite number of stages. For these types of problems, computations can be organized and carried out in a tabular form. With this structure, the optimization problem at each stage can usually be solved by total enumeration of all possible outcomes. More complex dynamic programming models may include probabilistic components, continuous decision variables, or an infinite number of stages. In cases where the optimization problem at each stage involves continuous decision variables, linear programming or calculus-based procedures may be needed to obtain an optimal solution.

SUMMARY

Dynamic programming is an attractive approach to problem solving when it is possible to break a large problem into interrelated smaller problems. The solution procedure then proceeds recursively, solving one of the smaller problems at each stage. Dynamic programming is not a specific algorithm, but rather an approach to problem solving. Thus, the recursive optimization may be carried out differently for different problems. In any case, it is almost always easier to solve a series of smaller problems than one large one. Through this process, dynamic programming obtains its power. The Management Science in Action, The EPA and Water Quality Management, describes how the EPA uses a dynamic programming model to establish seasonal discharge limits that protect water quality.

MANAGEMENT SCIENCE IN ACTION

THE EPA AND WATER QUALITY MANAGEMENT*

The U.S. Environmental Protection Agency (EPA) is an independent agency of the executive branch of the federal government. The EPA administers comprehensive environmental protection laws related to the following areas:

- Water pollution control, water quality, and drinking water
- Air pollution and radiation
- Pesticides and toxic substances
- Solid and hazardous waste, including emergency spill response and Superfund site remediation

The EPA administers programs designed to maintain acceptable water quality conditions for rivers and streams throughout the United States. To guard against polluted rivers and streams, the government requires companies to obtain a discharge permit from federal or state authorities before any form of pollutants can be discharged into a body of water. These permits specifically notify each discharger as to the amount of legally dischargeable waste that can be placed in the river or stream. The discharge limits are determined by ensuring that water quality criteria are met even in unusually dry seasons when the river or stream has a critically low-flow condition. Most often, this condition is based on the lowest flow recorded over the past 10 years. Ensuring that water quality is maintained under the low-flow conditions provides a high degree of reliability that the water quality criteria can be maintained throughout the year.

A goal of the EPA is to establish seasonal discharge limits that enable lower treatment costs while maintaining water quality standards at a prescribed level of reliability. These discharge limits are established by first determining the design stream flow for the body of water receiving the waste. The design stream flows for each season interact to determine the overall reliability that the annual water quality conditions will be maintained. The Municipal Environmental Research Laboratory in Cincinnati, Ohio, developed a dynamic programming model to determine design stream flows, which in turn could be used to establish seasonal waste discharge limits. The model chose the design stream flows that minimized treatment cost subject to a reliability constraint that the probability of no water quality violation was greater than a minimal acceptable probability. The model contained a stage for each season, and the reliability constraint established the state variability for the dynamic programming model. With the use of this dynamic programming model, the EPA is able to establish seasonal discharge limits that provide a minimum-cost treatment plan that maintains EPA water quality standards.

*Based on information provided by John Convery of the Environmental Protection Agency.

GLOSSARY

Dynamic programming An approach to problem solving that permits decomposing a large problem that may be difficult to solve into a number of interrelated smaller problems that are usually easier to solve.

Principle of optimality Regardless of the decisions made at previous stages, if the decision made at stage n is to be part of an overall optimal solution, the decision made at stage n must be optimal for all remaining stages.

Stages When a large problem is decomposed into a number of subproblems, the dynamic programming solution approach creates a stage to correspond to each of the subproblems.

Decision variable d_n A variable representing the possible decisions that can be made at stage n.

State variables x_n and x_{n-1} An input state variable x_n and an output state variable x_{n-1} together define the condition of the process at the beginning and end of stage n.

Stage transformation function $t_n(x_n, d_n)$ The rule or equation that relates the output state variable x_{n-1} for stage n to the input state variable x_n and the decision variable d_n.

Return function $r_n(x_n, d_n)$ A value (such as profit or loss) associated with making decision d_n at stage n for a specific value of the input state variable x_n.

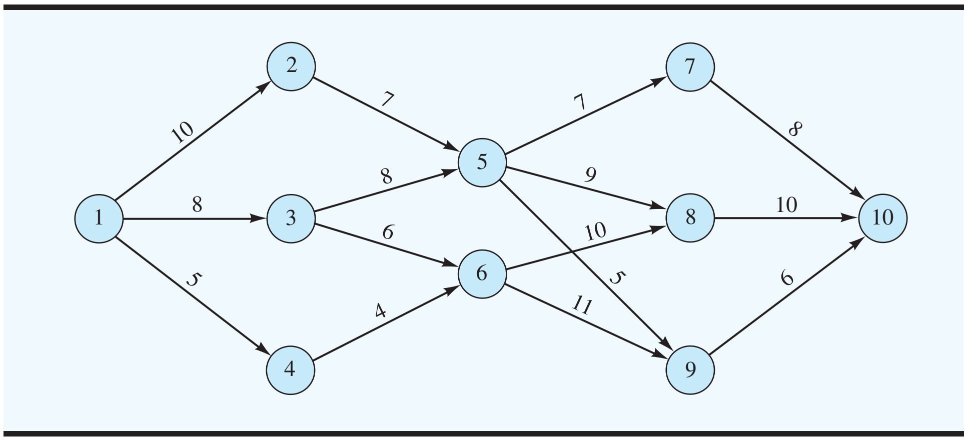
Knapsack problem Finding the number of N items, each of which has a different weight and value, that can be placed in a knapsack with limited weight capacity so as to maximize the total value of the items placed in the knapsack.

PROBLEMS

1. In Section 21.1 we solved a shortest-route problem using dynamic programming. Find the optimal solution to this problem by total enumeration; that is, list all 16 possible routes from the origin, node 1, to the destination, node 10, and pick the one with the smallest value. Explain why dynamic programming results in fewer computations for this problem.

SELF test

2. Consider the following network. The numbers above each arc represent the distance between the connected nodes.



a. Find the shortest route from node 1 to node 10 using dynamic programming.
b. What is the shortest route from node 4 to node 10?
c. Enumerate all possible routes from node 1 to node 10. Explain how dynamic programming reduces the number of computations to fewer than the number required by total enumeration.

SELF test

3. A charter pilot has additional capacity for 2000 pounds of cargo on a flight from Dallas to Seattle. A transport company has four types of cargo in Dallas to be delivered to Seattle. The number of units of each cargo type, the weight per unit, and the delivery fee per unit are shown.

Cargo Type	Units Available	Weight per Unit (100 pounds)	Delivery Fee ($100s)
1	2	8	22
2	2	5	12
3	4	3	7
4	3	2	3

a. Use dynamic programming to find how many units of each cargo type the pilot should contract to deliver.

b. Suppose the pilot agrees to take another passenger and the additional cargo capacity is reduced to 1800 pounds. How does your recommendation change?

4. A firm just hired eight new employees and would like to determine how to allocate their time to four activities. The firm prepared the following table, which gives the estimated profit for each activity as a function of the number of new employees allocated to it:

	Number of New Employees								
Activities	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	22	30	37	44	49	54	58	60	61
2	30	40	48	55	59	62	64	66	67
3	46	52	56	59	62	65	67	68	69
4	5	22	36	48	52	55	58	60	61

a. Use dynamic programming to determine the optimal allocation of new employees to the activities.

b. Suppose only six new employees were hired. Which activities would you assign to these employees?

5. A sawmill receives logs in 20-foot lengths, cuts them to smaller lengths, and then sells these smaller lengths to a number of manufacturing companies. The company has orders for the following lengths:

$$\begin{aligned} l_1 &= 3 \text{ ft} \\ l_2 &= 7 \text{ ft} \\ l_3 &= 11 \text{ ft} \\ l_4 &= 16 \text{ ft} \end{aligned}$$

The sawmill currently has an inventory of 2000 logs in 20-foot lengths and would like to select a cutting pattern that will maximize the profit made on this inventory. Assuming the sawmill has sufficient orders available, its problem becomes one of determining the cutting pattern that will maximize profits. The per-unit profit for each of the smaller lengths is as follows:

Length (feet)	3	7	11	16
Profit ($)	1	3	5	8

Any cutting pattern is permissible as long as

$$3d_1 + 7d_2 + 11d_3 + 16d_4 \leq 20$$

where d_i is the number of pieces of length l_i cut, $i = 1, 2, 3, 4$.

a. Set up a dynamic programming model of this problem, and solve it. What are your decision variables? What is your state variable?

b. Explain briefly how this model can be extended to find the best cutting pattern in cases where the overall length l can be cut into N lengths, $l_1, l_2, \ldots, l_N$.

6. A large manufacturing company has a well-developed management training program. Each trainee is expected to complete a four-phase program, but at each phase of the training program a trainee may be given a number of different assignments. The following assignments are available with their estimated completion times in months at each phase of the program.

Phase I	Phase II	Phase III	Phase IV
A–13	E–3	H–12	L–10
B–10	F–6	I–6	M–5
C–20	G–5	J–7	N–13
D–17		K–10	

Assignments made at subsequent phases depend on the previous assignment. For example, a trainee who completes assignment A at phase I may only go on to assignment F or G at phase II—that is, a precedence relationship exists for each assignment.

Assignment	Feasible Succeeding Assignments	Assignment	Feasible Succeeding Assignments
A	F, G	H	L, M
B	F	I	L, M
C	G	J	M, N
D	E, G	K	N
E	H, I, J, K	L	Finish
F	H, K	M	Finish
G	J, K	N	Finish

a. The company would like to determine the sequence of assignments that will minimize the time in the training program. Formulate and solve this problem as a dynamic programming problem. (*Hint:* Develop a network representation of the problem where each node represents completion of an activity.)

b. If a trainee just completed assignment F and would like to complete the remainder of the training program in the shortest possible time, which assignment should be chosen next?

7. Crazy Robin, the owner of a small chain of Robin Hood Sporting Goods stores in Des Moines and Cedar Rapids, Iowa, just purchased a new supply of 500 dozen top-line golf balls. Because she was willing to purchase the entire amount of a production overrun, Robin was able to buy the golf balls at one-half the usual price.

Three of Robin's stores do a good business in the sale of golf equipment and supplies, and, as a result, Robin decided to retail the balls at these three stores. Thus, Robin is faced with the problem of determining how many dozen balls to allocate to each store. The following estimates show the expected profit from allocating 100, 200, 300, 400, or 500 dozen to each store:

	Number of Dozens of Golf Balls				
Store	**100**	**200**	**300**	**400**	**500**
1	\$600	\$1100	\$1550	\$1700	\$1800
2	500	1200	1700	2000	2100
3	550	1100	1500	1850	1950

Assuming the lots cannot be broken into any sizes smaller than 100 dozen each, how many dozen golf balls should Crazy Robin send to each store?

8. The Max X. Posure Advertising Agency is conducting a 10-day advertising campaign for a local department store. The agency determined that the most effective campaign would possibly include placing ads in four media: daily newspaper, Sunday newspaper, radio, and television. A total of \$8000 has been made available for this campaign, and the agency would like to distribute this budget in \$1000 increments across the media in such a fashion that an advertising exposure index is maximized. Research conducted by the agency permits the following estimates to be made of the exposure per each \$1000 expenditure in each of the media.

	Thousands of Dollars Spent							
Media	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Daily newspaper	24	37	46	59	72	80	82	82
Sunday newspaper	15	55	70	75	90	95	95	95
Radio	20	30	45	55	60	62	63	63
Television	20	40	55	65	70	70	70	70

a. How much should the agency spend on each medium to maximize the department store's exposure?

b. How would your answer change if only \$6000 were budgeted?

c. How would your answers in parts (a) and (b) change if television were not considered as one of the media?

9. Suppose we have a three-stage process where the yield for each stage is a function of the decision made. In mathematical notation, we may state our problem as follows:

$$\begin{aligned} \text{Max} \quad & r_1(d_1) + r_2(d_2) + r_3(d_3) \\ \text{s.t.} \quad & \\ & d_1 + d_2 + d_3 \leq 1000 \end{aligned}$$

The possible values the decision variables may take on at each stage and the corresponding returns are as follows:

Stage 1		Stage 2		Stage 3	
d_1	$r_1(d_1)$	d_2	$r_2(d_2)$	d_3	$r_3(d_3)$
0	0	100	120	100	175
100	110	300	400	500	700
200	300	500	650		
300	400	600	700		
400	425	800	975		

a. Use total enumeration to list all feasible sequences of decisions for this problem. Which one is optimal [i.e., maximizes $r_1(d_1) + r_2(d_2) + r_3(d_3)$]?
b. Use dynamic programming to solve this problem.

SELF test

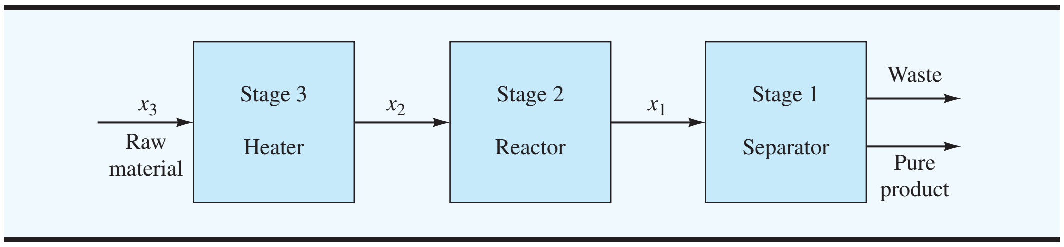
10. Recall the production and inventory control problem of Section 21.4. Mills Manufacturing Company has just such a production and inventory control problem for an armature the company manufactures as a component for a generator. The available data for the next 3-month planning period are as follow:

		Capacity		Cost per Unit	
Month	**Demand**	**Production**	**Warehouse**	**Production**	**Holding**
1	20	30	40	$2.00	$0.30
2	30	20	30	1.50	0.30
3	30	30	20	2.00	0.20

Use dynamic programming to find the optimal production quantities and inventory levels in each period for the Mills Manufacturing Company. Assume an inventory of 10 units on hand at the beginning of month 1 and production runs are completed in multiples of 10 units (i.e., 10, 20, or 30 units).

Case Problem PROCESS DESIGN

The Baker Chemical processing plant is considering introducing a new product. However, before making a final decision, management requests estimates of profits associated with different process designs. The general flow process is represented:



Raw material is fed into a heater at the rate of 4500 pounds per week. The heated material is routed to a reactor where a portion of the raw material is converted to pure product. A separator then withdraws the finished product for sale. The unconverted material is discarded as waste.

Profit considerations are to be based on a two-year payback period on investments; that is, all capital expenditures must be recovered in two years (100 weeks). All calculations will be based on weekly operations. Raw material costs are expected to stay fixed at $1 per pound, and the forecasted selling price for the finished product is $6 per pound.

It is your responsibility to determine the process design that will yield maximum profit per week. You and your coworkers collect the following preliminary data.

One heater with an initial cost of \$12,000 is being considered at stage 3. Two temperatures, 700°F and 800°F, are feasible. The operating costs for the heater depend directly on the temperature to be attained. These costs are as follows:

Operating Costs at Stage 3

	Decisions at Stage 3	
Input x_3	**700°F**	**800°F**
4500 lbs.	\$280/week	\$380/week

Stage 3's output x_2, which is also the input to stage 2, may be expressed as 4500 pounds of raw material heated to either 700°F or 800°F. One of the decisions you must make is to choose the temperature for heating the raw material.

A reactor, which can operate with either of two catalysts, C1 or C2, is to be used for stage 2. The initial cost of this reactor is \$50,000. The operating costs of this reactor are independent of the input x_2 and depend only on the catalyst selected. The costs of the catalysts are included in the operating costs. The output will be expressed in pounds of converted (or pure) material. The percentage of material converted depends on the incoming temperature and the catalyst used. The following tables summarize the pertinent information. Thus, a second decision you must make is to specify which catalyst should be used.

Percent Conversion

	Decisions at Stage 2	
x_2	**C1**	**C2**
(4500 lbs., 700°F)	20	40
(4500 lbs., 800°F)	40	60

Operating Costs
Decisions at Stage 2

C1	**C2**
\$450/week	\$650/week

One of two separators, S1 or S2, will be purchased for stage 1. The S1 separator has an initial cost of \$20,000 and a weekly operating cost of \$0.10 per pound of pure product to be separated. Comparatively, S2 has an initial cost of \$5000 and a weekly operating cost of \$0.20. Included in these operating costs is the expense of discarding the unconverted raw material as waste.

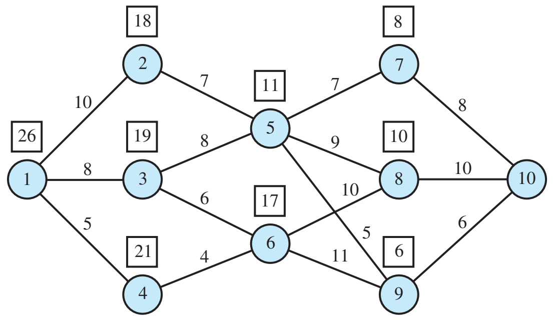
Managerial Report

1. Develop a dynamic programming model for the Baker Chemical process design.
2. Make specific recommendations on the following:
   - Best temperature for the heater
   - Best catalyst to use with the reactor
   - Best separator to purchase
3. What is the weekly profit?

Self-Test Solutions and Answers to Even-Numbered Problems

Chapter 21

2. a. The numbers in the squares above each node represent the shortest route from the node to node 10



The shortest route is given by the sequence of nodes (1–4–6–9–10)

b. The shortest route from node 4 to node 10 is given by (4–6–9–10)

c.

Route	Value	Route	Value
(1–2–5–7–10)	32	(1–3–6–8–10)	34
(1–2–5–8–10)	36	(1–3–6–9–10)	31
(1–2–5–9–10)	28	(1–4–6–8–10)	29
(1–3–5–7–10)	31	(1–4–6–9–10)	26
(1–3–5–8–10)	35		
(1–3–5–9–10)	27		

3. Use four stages (one for each type of cargo); let the state variable represent the amount of cargo space remaining

a. In hundreds of pounds, we have up to 20 units of capacity available

Stage 1 (Cargo Type 1):

x_1	0	1	2	d_1^*	$f_1(x_1)$	x_0
0–7	0	—	—	0	0	0–7
8–15	0	22	—	1	22	0–7
16–20	0	22	44	2	44	0–4

Stage 2 (Cargo Type 2):

x_2	0	1	2	d_2^*	$f_2(x_2)$	x_1
0–4	0	—	—	0	0	0–4
5–7	0	12	—	1	12	0–2
8–9	22	12	—	0	22	8–9
10–12	22	12	24	2	24	0–2
13–15	22	34	24	1	34	8–10
16–17	44	34	24	0	44	16–17
18–20	44	34	46	2	46	8–10

Stage 3 (Cargo Type 3):

x_3	0	1	2	3	4	d_3^*	$f_3(x_3)$	x_2
0–2	0	—	—	—	—	0	0	0–2
3–4	0	7	—	—	—	1	7	0–1
5	12	7	—	—	—	0	12	5
6–7	12	7	14	—	—	2	14	0–1
8	22	19	14	—	—	0	22	8
9	22	19	14	21	—	0	22	9
10	24	19	14	21	—	0	24	10
11	24	29	26	21	—	1	29	8
12	24	29	26	21	28	1	29	9
13	34	31	26	21	28	0	34	13
14–15	34	31	36	33	28	2	36	8–9
16	44	41	38	33	28	0	44	16
17	44	41	38	43	40	0	44	17
18	46	41	38	43	40	0	46	18
19	46	51	48	45	40	1	51	16
20	46	51	48	45	50	1	51	17

Stage 4 (Cargo Type 4):

x_4	0	1	2	3	d_4^*	$f_4(x_4)$	x_3
20	51	49	50	45	0	51	20

Tracing back through the tables, we find

Stage	State Variable Entering	Optimal Decision	State Variable Leaving
4	20	0	20
3	20	1	17
2	17	0	17
1	17	2	1

Load 1 unit of cargo type 3 and 2 units of cargo type 1 for a total return of \$5100

b. Only the calculations for stage 4 need to be repeated; the entering value for the state variable is 18

x_4	0	1	2	3	d_4^*	$f_4(x_4)$	x_3
18	46	47	42	38	1	47	16

Optimal solution: $d_4 = 1, d_3 = 0, d_2 = 0, d_1 = 2$
Value = 47

4. a. Alternative optimal solutions: value = 186
Solution 1: A1–3, A2–2, A3–0, A4–3
Solution 2: A1–2, A2–3, A3–0, A4–3

b. Value = 172; A1–1, A2–2, A3–0, A4–3

6. a. A–G–J–M

b. Choose H

8. a. Daily news—1, Sunday news—3, radio—1, TV—3; Max exposure = 169

b. 1, 2, 1, 2; Max exposure = 139

c. For part (a): 2, 3, 3; Max exposure = 152
For part (b): 2, 3, 1; Max exposure = 127

10. The optimal production schedule is as follows:

Month	Beginning Inventory	Production	Ending Inventory
1	10	20	10
2	10	20	0
3	0	30	0