Chapter 10

Deterministic Dynamic Programming

Chapter Guide. Dynamic programming (DP) determines the optimum solution of a multivariable problem by decomposing it into stages, each stage comprising a single-variable subproblem. The advantage of the decomposition is that the optimization process at each stage involves one variable only, a simpler task computationally than dealing with all the variables simultaneously. A DP model is basically a recursive equation linking the different stages of the problem in a manner that guarantees that each stage's optimal feasible solution is also optimal and feasible for the entire problem. The notation and the conceptual framework of the recursive equation are unlike any you have studied so far. Experience has shown that the structure of the recursive equation may not appear "logical" to a beginner. Should you have a similar experience, the best course of action is to try to implement what may appear logical to you, and then carry out the computations accordingly. You will soon discover that the definitions in the book are the correct ones and, in the process, will learn how DP works. We have also included two partially automated Excel spreadsheets for some of the examples in which the user must provide key information to drive the DP computations. The exercise should help you understand some of the subtleties of DP.

Although the recursive equation is a common framework for formulating DP models, the solution details differ. Only through exposure to different formulations will you be able to gain experience in DP modeling and DP solution. A number of deterministic DP applications are given in this chapter. Chapter 22 on the CD presents probabilistic DP applications. Other applications in the important area of inventory modeling are presented in Chapters 11 and 14.

This chapter includes a summary of 1 real-life application, 7 solved examples, 2 Excel spreadsheet models, 32 end-of-section problems, and 1 case. The case is in Appendix E on the CD. The AMPL/Excel/Solver/TORA programs are in folder ch10Files.
Real-Life Application—Optimization of Crosscutting and Log Allocation at Weyerhaeuser.

Mature trees are harvested and crosscut into logs to manufacture different end products (such as construction lumber, plywood, wafer boards, or paper). Log specifications (e.g., length and end diameters) differ depending on the mill where the logs are used. With harvested trees measuring up to 100 feet in length, the number of crosscut combinations meeting mill requirements can be large, and the manner in which a tree is disassembled into logs can affect revenues. The objective is to determine the crosscut combinations that maximize the total revenue. The study uses dynamic programming to optimize the process. The proposed system was first implemented in 1978 with an annual increase in profit of at least $7 million. Case 8 in Chapter 24 on the CD provides the details of the study.

10.1 RECURSIVE NATURE OF COMPUTATIONS IN DP

Computations in DP are done recursively, so that the optimum solution of one subproblem is used as an input to the next subproblem. By the time the last subproblem is solved, the optimum solution for the entire problem is at hand. The manner in which the recursive computations are carried out depends on how we decompose the original problem. In particular, the subproblems are normally linked by common constraints. As we move from one subproblem to the next, the feasibility of these common constraints must be maintained.

Example 10.1-1 (Shortest-Route Problem)

Suppose that you want to select the shortest highway route between two cities. The network in Figure 10.1 provides the possible routes between the starting city at node 1 and the destination city at node 7. The routes pass through intermediate cities designated by nodes 2 to 6.

To solve the problem by DP, we first decompose it into stages as delineated by the vertical dashed lines in Figure 10.2. Next, we carry out the computations for each stage separately.

FIGURE 10.1
Route network for Example 10.1-1
The general idea for determining the shortest route is to compute the shortest (cumulative) distances to all the terminal nodes of a stage and then use these distances as input data to the immediately succeeding stage. Starting from node 1, stage 1 includes three end nodes (2, 3, and 4) and its computations are simple.

**Stage 1 Summary.**

- Shortest distance from node 1 to node 2 = 7 miles \( (\text{from node 1}) \)
- Shortest distance from node 1 to node 3 = 8 miles \( (\text{from node 1}) \)
- Shortest distance from node 1 to node 4 = 5 miles \( (\text{from node 1}) \)

Next, stage 2 has two end nodes, 5 and 6. Considering node 5 first, we see from Figure 10.2 that node 5 can be reached from three nodes, 2, 3, and 4, by three different routes: \((2, 5)\), \((3, 5)\), and \((4, 5)\). This information, together with the shortest distances to nodes 2, 3, and 4, determines the shortest (cumulative) distance to node 5 as

\[
\text{Shortest distance to node 5} = \min_{i=2,3,4} \left( \text{Shortest distance to node } i \right) + \left( \text{Distance from node } i \text{ to node } 5 \right)
\]

\[
= \min \left\{ \begin{array}{l}
7 + 12 = 19 \\
8 + 8 = 16 \\
5 + 7 = 12
\end{array} \right\} = 12 \quad (\text{from node 4})
\]

Node 6 can be reached from nodes 3 and 4 only. Thus

\[
\text{Shortest distance to node 6} = \min_{i=3,4} \left( \text{Shortest distance to node } i \right) + \left( \text{Distance from node } i \text{ to node } 6 \right)
\]

\[
= \min \left\{ \begin{array}{l}
8 + 9 = 17 \\
5 + 13 = 18
\end{array} \right\} = 17 \quad (\text{from node 3})
\]
Stage 2 Summary.

Shortest distance from node 1 to node 5 = 12 miles (from node 4)
Shortest distance from node 1 to node 6 = 17 miles (from node 3)

The last step is to consider stage 3. The destination node 7 can be reached from either nodes 5 or 6. Using the summary results from stage 2 and the distances from nodes 5 and 6 to node 7, we get

\[
\text{Shortest distance to node 7} = \min_{i=5,6} \left( \text{Shortest distance to node } i + \text{Distance from } i \text{ to node 7} \right)
\]

\[
= \min \{12 + 9 = 21, 17 + 6 = 23\} = 21 \text{ (from node 5)}
\]

Stage 3 Summary.

Shortest distance from node 1 to node 7 = 21 miles (from node 5)

Stage 3 summary shows that the shortest distance between nodes 1 and 7 is 21 miles. To determine the optimal route, stage 3 summary links node 7 to node 5, stage 2 summary links node 4 to node 5, and stage 1 summary links node 4 to node 1. Thus, the shortest route is 1 → 4 → 5 → 7.

The example reveals the basic properties of computations in DP:

1. The computations at each stage are a function of the feasible routes of that stage, and that stage alone.
2. A current stage is linked to the immediately preceding stage only without regard to earlier stages. The linkage is in the form of the shortest-distance summary that represents the output of the immediately preceding stage.

Recursive Equation. We now show how the recursive computations in Example 10.1-1 can be expressed mathematically. Let \( f_i(x_i) \) be the shortest distance to node \( x_i \) at stage \( i \), and define \( d(x_{i-1}, x_i) \) as the distance from node \( x_{i-1} \) to node \( x_i \); then \( f_i \) is computed from \( f_{i-1} \) using the following recursive equation:

\[
f_i(x_i) = \min_{\text{all} \ (x_{i-1}, x_i) \text{ routes}} \{d(x_{i-1}, x_i) + f_{i-1}(x_{i-1})\}, \ i = 1, 2, 3
\]

Starting at \( i = 1 \), the recursion sets \( f_0(x_0) = 0 \). The equation shows that the shortest distances \( f_i(x_i) \) at stage \( i \) must be expressed in terms of the next node, \( x_i \). In the DP terminology, \( x_i \) is referred to as the state of the system at stage \( i \). In effect, the state of the system at stage \( i \) is the information that links the stages together, so that optimal decisions for the remaining stages can be made without reexamining how the decisions for the previous stages are reached. The proper definition of the state allows us to consider each stage separately and guarantee that the solution is feasible for all the stages.

The definition of the state leads to the following unifying framework for DP.
10.2 Forward and Backward Recursion

**Principle of Optimality**

Future decisions for the remaining stages will constitute an optimal policy regardless of the policy adopted in previous stages.

The implementation of the principle is evident in the computations in Example 10.1-1. For example, in stage 3, we only use the shortest distances to nodes 5 and 6, and do not concern ourselves with how these nodes are reached from node 1. Although the principle of optimality is “vague” about the details of how each stage is optimized, its application greatly facilitates the solution of many complex problems.

**PROBLEM SET 10.1A**

*1. Solve Example 10.1-1, assuming the following routes are used:

\[d(1, 2) = 5, d(1, 3) = 9, d(1, 4) = 8\]
\[d(2, 5) = 10, d(2, 6) = 17\]
\[d(3, 5) = 4, d(3, 6) = 10\]
\[d(4, 5) = 9, d(4, 6) = 9\]
\[d(5, 7) = 8\]
\[d(6, 7) = 9\]

2. I am an avid hiker. Last summer, I went with my friend G. Don on a 5-day hike-and-camp trip in the beautiful White Mountains in New Hampshire. We decided to limit our hiking to an area comprising three well-known peaks: Mounts Washington, Jefferson, and Adams. Mount Washington has a 6-mile base-to-peak trail. The corresponding base-to-peak trails for Mounts Jefferson and Adams are 4 and 5 miles, respectively. The trails joining the bases of the three mountains are 3 miles between Mounts Washington and Jefferson, 2 miles between Mounts Jefferson and Adams, and 5 miles between Mounts Adams and Washington. We started on the first day at the base of Mount Washington and returned to the same spot at the end of 5 days. Our goal was to hike as many miles as we could. We also decided to climb exactly one mountain each day and to camp at the base of the mountain we would be climbing the next day. Additionally, we decided that the same mountain could not be visited in any two consecutive days. How did we schedule our hike?

FORWARD AND BACKWARD RECURSION

Example 10.1-1 uses forward recursion in which the computations proceed from stage 1 to stage 3. The same example can be solved by backward recursion, starting at stage 3 and ending at stage 1.

Both the forward and backward recursions yield the same solution. Although the forward procedure appears more logical, DP literature invariably uses backward recursion. The reason for this preference is that, in general, backward recursion may be more efficient computationally. We will demonstrate the use of backward recursion by applying it to Example 10.1-1. The demonstration will also provide the opportunity to present the DP computations in a compact tabular form.
Example 10.2-1

The backward recursive equation for Example 10.2-1 is

\[ f_i(x_i) = \min_{\text{all routes } (x_i, x_{i+1})} \{d(x_i, x_{i+1}) + f_{i+1}(x_{i+1})\}, \quad i = 1, 2, 3 \]

where \( f_4(x_4) = 0 \) for \( x_4 = 7 \). The associated order of computations is \( f_3 \rightarrow f_2 \rightarrow f_1 \).

Stage 3. Because node 7 (\( x_4 = 7 \)) is connected to nodes 5 and 6 (\( x_3 = 5 \) and 6) with exactly one route each, there are no alternatives to choose from, and stage 3 results can be summarized as

<table>
<thead>
<tr>
<th>( x_3 )</th>
<th>( x_4 = 7 )</th>
<th>( f_3(x_3) )</th>
<th>( x_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>9</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Stage 2. Route (2, 6) is blocked because it does not exist. Given \( f_3(x_3) \) from stage 3, we can compare the feasible alternatives as shown in the following tableau:

<table>
<thead>
<tr>
<th>( x_2 )</th>
<th>( x_3 = 5 )</th>
<th>( x_3 = 6 )</th>
<th>( f_3(x_2) )</th>
<th>( x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12 + 9 = 21</td>
<td>-</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>8 + 9 = 17</td>
<td>9 + 6 = 15</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>7 + 9 = 16</td>
<td>13 + 6 = 19</td>
<td>16</td>
<td>5</td>
</tr>
</tbody>
</table>

The optimum solution of stage 2 reads as follows: If you are in cities 2 or 4, the shortest route passes through city 5, and if you are in city 3, the shortest route passes through city 6.

Stage 1. From node 1, we have three alternative routes: (1, 2), (1, 3), and (1, 4). Using \( f_2(x_2) \) from stage 2, we can compute the following tableau:

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 = 2 )</th>
<th>( x_2 = 3 )</th>
<th>( x_2 = 4 )</th>
<th>( f_3(x_1) )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7 + 21 = 28</td>
<td>8 + 15 = 23</td>
<td>5 + 16 = 21</td>
<td>21</td>
<td>4</td>
</tr>
</tbody>
</table>

The optimum solution at stage 1 shows that city 1 is linked to city 4. Next, the optimum solution at stage 2 links city 4 to city 5. Finally, the optimum solution at stage 3 connects city 5 to city 7. Thus, the complete route is given as 1 \( \rightarrow \) 4 \( \rightarrow \) 5 \( \rightarrow \) 7, and the associated distance is 21 miles.

PROBLEM SET 10.2A

1. For Problem 1, Set 10.1a, develop the backward recursive equation, and use it to find the optimum solution.
2. For Problem 2, Set 10.1a, develop the backward recursive equation, and use it to find the optimum solution.

3. For the network in Figure 10.3, it is desired to determine the shortest route between cities 1 to 7. Define the stages and the states using backward recursion, and then solve the problem.

10.3 SELECTED DP APPLICATIONS

This section presents four applications, each with a new idea in the implementation of dynamic programming. As you study each application, pay special attention to the three basic elements of the DP model:

1. Definition of the stages
2. Definition of the alternatives at each stage
3. Definition of the states for each stage

Of the three elements, the definition of the state is usually the most subtle. The applications presented here show that the definition of the state varies depending on the situation being modeled. Nevertheless, as you investigate each application, you will find it helpful to consider the following questions:

1. What relationships bind the stages together?
2. What information is needed to make feasible decisions at the current stage without reexamining the decisions made at previous stages?

My teaching experience indicates that understanding the concept of the state can be enhanced by questioning the validity of the way it is defined in the book. Try a different definition that may appear "more logical" to you, and use it in the recursive computations. You will eventually discover that the definitions presented here provide the correct way for solving the problem. Meanwhile, the proposed mental process should enhance your understanding of the concept of the state.

10.3.1 Knapsack/Fly-Away/Cargo-Loading Model

The knapsack model classically deals with the situation in which a soldier (or a hiker) must decide on the most valuable items to carry in a backpack. The problem paraphrases
a general resource allocation model in which a single limited resource is assigned to a number of alternatives (e.g., limited funds assigned to projects) with the objective of maximizing the total return.

Before presenting the DP model, we remark that the knapsack problem is also known in the literature as the fly-away kit problem, in which a jet pilot must determine the most valuable (emergency) items to take aboard a jet; and the cargo-loading problem, in which a vessel with limited volume or weight capacity is loaded with the most valuable cargo items. It appears that the three names were coined to ensure equal representation of three branches of the armed forces: Air Force, Army, and Navy!

The (backward) recursive equation is developed for the general problem of an n-item W-lb knapsack. Let $m_i$ be the number of units of item $i$ in the knapsack and define $r_i$ and $w_i$ as the revenue and weight per unit of item $i$. The general problem is represented by the following ILP:

$$\text{Maximize } z = r_1 m_1 + r_2 m_2 + \cdots + r_n m_n$$

subject to

$$w_1 m_1 + w_2 m_2 + \cdots + w_n m_n \leq W$$
$$m_1, m_2, \ldots, m_n \geq 0 \text{ and integer}$$

The three elements of the model are

1. **Stage $i$** is represented by item $i$, $i = 1, 2, \ldots, n$.
2. The **alternatives** at stage $i$ are represented by $m_i$, the number of units of item $i$ included in the knapsack. The associated return is $r_i m_i$. Defining $\left[\frac{W}{w_i}\right]$ as the largest integer less than or equal to $\frac{W}{w_i}$, it follows that $m_i = 0, 1, \ldots, \left[\frac{W}{w_i}\right]$.
3. The **state** at stage $i$ is represented by $x_i$, the total weight assigned to stages (items) $i, i + 1, \ldots, n$. This definition reflects the fact that the weight constraint is the only restriction that links all $n$ stages together.

Define

$$f_i(x_i) = \text{maximum return for stages } i, i + 1, \text{ and } n, \text{ given state } x_i$$

The simplest way to determine a recursive equation is a two-step procedure:

**Step 1.** Express $f_i(x_i)$ as a function of $f_i(x_{i+1})$ as follows:

$$f_i(x_i) = \min_{m_i=0,1,\ldots,\left[\frac{W}{w_i}\right]} \{r_i m_i + f_{i+1}(x_{i+1})\}, i = 1, 2, \ldots, n$$

$$f_{n+1}(x_{n+1}) = 0$$

**Step 2.** Express $x_{i+1}$ as a function of $x_i$ to ensure that the left-hand side, $f_i(x_i)$, is a function of $x_i$ only. By definition, $x_i - x_{i+1} = w_i m_i$ represents the weight used at stage $i$. Thus, $x_{i+1} = x_i - w_i m_i$, and the proper recursive equation is given as

$$f_i(x_i) = \max_{m_i=0,1,\ldots,\left[\frac{W}{w_i}\right]} \{r_i m_i + f_{i+1}(x_i - w_i m_i)\}, i = 1, 2, \ldots, n$$
Example 10.3-1

A 4-ton vessel can be loaded with one or more of three items. The following table gives the unit weight, \( w_i \) in tons and the unit revenue in thousands of dollars, \( r_i \), for item \( i \). How should the vessel be loaded to maximize the total return?

<table>
<thead>
<tr>
<th>( i )</th>
<th>( w_i )</th>
<th>( r_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>

Because the unit weights \( w_i \) and the maximum weight \( W \) are integer, the state \( x_i \) assumes integer values only.

**Stage 3.** The exact weight to be allocated to stage 3 (item 3) is not known in advance, but can assume one of the values 0, 1, ..., and 4 (because \( W = 4 \) tons). The states \( x_3 = 0 \) and \( x_3 = 4 \), respectively, represent the extreme cases of not shipping item 3 at all and of allocating the entire vessel to it. The remaining values of \( x_3 (= 1, 2, \text{ and } 3) \) imply a partial allocation of the vessel capacity to item 3. In effect, the given range of values for \( x_3 \) covers all possible allocations of the vessel capacity to item 3.

Given \( w_3 = 1 \) ton per unit, the maximum number of units of item 3 that can be loaded is \( \frac{4}{w_3} = 4 \), which means that the possible values of \( m_3 \) are 0, 1, 2, 3, and 4. An alternative \( m_3 \) is feasible only if \( w_3 m_3 \leq x_3 \). Thus, all the infeasible alternatives (those for which \( w_3 m_3 > x_3 \)) are excluded. The following equation is the basis for comparing the alternatives of stage 3.

\[
f_3(x_3) = \max_{m_3 = 0,1,\ldots,4} \{14m_3\}
\]

The following tableau compares the feasible alternatives for each value of \( x_3 \).

<table>
<thead>
<tr>
<th>( x_3 )</th>
<th>( m_3 = 0 )</th>
<th>( m_3 = 1 )</th>
<th>( m_3 = 2 )</th>
<th>( m_3 = 3 )</th>
<th>( m_3 = 4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>14</td>
<td>28</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>14</td>
<td>28</td>
<td>42</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>14</td>
<td>28</td>
<td>42</td>
<td>56</td>
</tr>
</tbody>
</table>

**Optimum solution**

<table>
<thead>
<tr>
<th>( m_3^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

**Stage 2.** \( \max \{m_3\} = \left[ \frac{4}{1} \right] = 1 \), or \( m_3 = 0, 1 \)

\[
f_2(x_2) = \max_{m_2 = 0,1} \{47m_2 + f_3(x_2 - 3m_2)\}
\]

<table>
<thead>
<tr>
<th>( x_2 )</th>
<th>( m_2 = 0 )</th>
<th>( m_2 = 1 )</th>
<th>( m_3^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 + 0 = 0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 + 14 = 14</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>0 + 28 = 28</td>
<td>-</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>0 + 42 = 42</td>
<td>47 + 0 = 47</td>
<td>47</td>
</tr>
<tr>
<td>4</td>
<td>0 + 56 = 56</td>
<td>47 + 14 = 61</td>
<td>61</td>
</tr>
</tbody>
</table>
Stage 1. \[ \max \{ m_1 = \frac{3}{2} \} = 2 \text{ or } m_1 = 0, 1, 2 \]

\[ f_1(x_1) = \max \{ 31m_1 + f_2(x_1 - 2m_1) \}, \max \{ m_1 = \frac{3}{2} \} = 2 \]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( m_1 = 0 )</th>
<th>( m_1 = 1 )</th>
<th>( m_1 = 2 )</th>
<th>( f_1(x_1) )</th>
<th>( m^*_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 + 0 = 0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 + 14 = 14</td>
<td>-</td>
<td>-</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0 + 28 = 28</td>
<td>31 + 0 = 31</td>
<td>-</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0 + 47 = 47</td>
<td>31 + 14 = 45</td>
<td>-</td>
<td>47</td>
<td>0</td>
</tr>
</tbody>
</table>

The optimum solution is determined in the following manner: Given \( W = 4 \) tons, from stage 1, \( x_1 = 4 \) gives the optimum alternative \( m^*_1 = 2 \), which means that 2 units of item 1 will be loaded on the vessel. This allocation leaves \( x_2 = x_1 - 2m^*_2 = 4 - 2 \times 2 = 0 \). From stage 2, \( x_2 = 0 \) yields \( m^*_2 = 0 \), which, in turn, gives \( x_3 = x_2 - 3m^*_2 = 0 - 3 \times 0 = 0 \). Next, from stage 3, \( x_3 = 0 \) gives \( m^*_3 = 0 \). Thus, the complete optimal solution is \( m^*_1 = 2, m^*_2 = 0, \) and \( m^*_3 = 0 \). The associated return is \( f_1(4) = 62,000 \).

In the table for stage 1, we actually need to obtain the optimum for \( x_1 = 4 \) only because this is the last stage to be considered. However, the computations for \( x_1 = 0, 1, 2, \) and 3 are included to allow carrying out sensitivity analysis. For example, what happens if the vessel capacity is 3 tons in place of 4 tons? The new optimum solution can be determined as

\[ (x_1 = 3) \rightarrow (m^*_1 = 0) \rightarrow (x_2 = 3) \rightarrow (m^*_2 = 1) \rightarrow (x_3 = 0) \rightarrow (m^*_3 = 0) \]

Thus the optimum is \((m^*_1, m^*_2, m^*_3) = (0, 1, 0)\) and the optimum revenue is \( f_1(3) = 47,000 \).

Remarks. The cargo-loading example represents a typical resource allocation model in which a limited resource is apportioned among a finite number of (economic) activities. The objective maximizes an associated return function. In such models, the definition of the state at each stage will be similar to the definition given for the cargo-loading model. Namely, the state at stage \( i \) is the total resource amount allocated to stages \( i, i + 1, \ldots, \) and \( n \).

Excel moment

The nature of dynamic programming computations makes it impossible to develop a general computer code that can handle all DP problems. Perhaps this explains the persistent absence of commercial DP software.

In this section, we present an Excel-based algorithm for handling a subclass of DP problems: the single-constraint knapsack problem (file Knapsack.xls). The algorithm is not data specific and can handle problems in this category with 10 alternatives or less.

Figure 10.4 shows the starting screen of the knapsack (backward) DP model. The screen is divided into two sections: The right section (columns Q:V) is used to summarize...
the output solution. In the left section (columns A:P), rows 3, 4, and 6 provide the input data for the current stage, and rows 7 and down are reserved for stage computations. The input data symbols correspond to the mathematical notation in the DP model, and are self-explanatory. To fit the spreadsheet conveniently on one screen, the maximum feasible value for alternative $m_i$ at stage $i$ is 10 (cells D6:N6).

Figure 10.5 shows the stage computations generated by the algorithm for Example 10.3-1. The computations are carried out one stage at a time, and the user provides the basic data that drive each stage. Engaging you in this manner will enhance your understanding of the computational details in DP.

Starting with stage 3, and using the notation and data in Example 10.3-1, the input cells are updated as the following list shows:

<table>
<thead>
<tr>
<th>Cell(s)</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3</td>
<td>Number of stages, $N = 3$</td>
</tr>
<tr>
<td>G3</td>
<td>Resource limit, $W = 4$</td>
</tr>
<tr>
<td>C4</td>
<td>Current stage = 3</td>
</tr>
<tr>
<td>E4</td>
<td>$w_3 = 1$</td>
</tr>
<tr>
<td>G4</td>
<td>$r_3 = 14$</td>
</tr>
<tr>
<td>D6:H6</td>
<td>$m_3 = (0, 1, 2, 3, 4)$</td>
</tr>
</tbody>
</table>

Note that the feasible values of $m_2$ are $0, 1, \ldots$, and $\left\lfloor \frac{W}{w_2} \right\rfloor = \left\lfloor \frac{4}{1} \right\rfloor = 4$, as in Example 10.3-1. The spreadsheet automatically tells you how many $m_2$-values are needed and checks the validity of the values you enter by issuing self-explanatory messages in row 5: "yes," "no," and "delete."

As stage 3 data are entered and verified, the spreadsheet will "come alive" and will generate all the necessary computations of the stage (columns B through P) automatically. The value $-1111111$ is used to indicate that the corresponding entry is not feasible. The optimum solution $(f_3, m_3)$ for the stage is given in columns O and P. Column A provides the values of $f_4$. Because the computations start at stage 3, $f_4 = 0$ for all values of $x_3$. You can leave A9:A13 blank or enter all zero values.
FIGURE 10.5
Excel DP model for the knapsack problem of Example 10.3-1 (file excelKnapsack.xls)

Now that stage 3 calculations are at hand, take the following steps to create a permanent record of the optimal solution of the current stage and to prepare the spreadsheet for next stage calculations:

Step 1. Copy the \(x_3\)-values, C9:C13, and paste them in Q5:Q9 in the optimum solution section. Next, copy the \((f_3, m_3)\)-values, O9:P13, and paste them in R5:S9. Remember that you need to paste values only, which requires selecting Paste Special from Edit menu and Values from the dialogue box.

Step 2. Copy the \(h\)-values in R5:R9 and paste them in A9:A13 (you do not need Paste Special in this step).

Step 3. Change cell C4 to 2 and enter the new values of \(w_2\), \(r_2\), and \(m_2\) to record the data of stage 2.

Step 2 places \(f_{i+1}(x_i - w_im_i)\) in column A in preparation for calculating \(f_i(x_i)\) at stage \(i\) (see the recursive formula for the knapsack problem in Example 10.3-1). This explains the reason for entering zero values, representing \(f_4\), in column A of stage 3 tableau.
Once stage 2 computations are available, you can prepare the screen for stage 1 in a similar manner. When stage 1 is complete, the optimum solution summary can be used to read the solution, as was explained in Example 10.3-1. Note that the organization of the output solution summary area (right section of the screen, columns Q:V) is free-formatted and you can organize its contents in any convenient manner you desire.

PROBLEM SET 10.3A

1. In Example 10.3-1, determine the optimum solution, assuming that the maximum weight capacity of the vessel is 2 tons then 5 tons.

2. Solve the cargo-loading problem of Example 10.3-1 for each of the following sets of data:
   *(a) $w_1 = 4, r_1 = 70, w_2 = 1, r_2 = 20, w_3 = 2, r_3 = 40, W = 6$
   *(b) $w_1 = 1, r_1 = 30, w_2 = 2, r_2 = 60, w_3 = 3, r_3 = 80, W = 4$

3. In the cargo-loading model of Example 10.3-1, suppose that the revenue per item includes a constant amount that is realized only if the item is chosen, as the following table shows:

<table>
<thead>
<tr>
<th>Item</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{-5 + 3m_1, \quad \text{if } m_1 &gt; 0}</td>
</tr>
<tr>
<td></td>
<td>{0, \quad \text{otherwise}}</td>
</tr>
<tr>
<td>2</td>
<td>{-15 + 47m_2, \quad \text{if } m_2 &gt; 0}</td>
</tr>
<tr>
<td></td>
<td>{0, \quad \text{otherwise}}</td>
</tr>
<tr>
<td>3</td>
<td>{-4 + 14m_3, \quad \text{if } m_3 &gt; 0}</td>
</tr>
<tr>
<td></td>
<td>{0, \quad \text{otherwise}}</td>
</tr>
</tbody>
</table>

   Find the optimal solution using DP. *(Hint: You can use the Excel file excelSetupKnapsack.xls to check your calculations.)*

4. A wilderness hiker must pack three items: food, first-aid kits, and clothes. The backpack has a capacity of 3 ft$^3$. Each unit of food takes 1 ft$^3$. A first-aid kit occupies $\frac{1}{4}$ ft$^3$ and each piece of cloth takes about $\frac{1}{6}$ ft$^3$. The hiker assigns the priority weights 3, 4, and 5 to food, first aid, and clothes, which means that clothes are the most valuable of the three items. From experience, the hiker must take at least one unit of each item and no more than two first-aid kits. How many of each item should the hiker take?

5. A student must select 10 electives from four different departments, with at least one course from each department. The 10 courses are allocated to the four departments in a manner that maximizes "knowledge." The student measures knowledge on a 100-point scale and comes up with the following chart:

<table>
<thead>
<tr>
<th>No. of courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department 1</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>100</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>

   How should the student select the courses?

---

*In this Problem Set, you are encouraged where applicable to work out the computations by hand and then verify the results using the template excelKnapsack.xls.*
6. I have a small backyard garden that measures $10 \times 20$ feet. This spring I plan to plant three types of vegetables: tomatoes, green beans, and corn. The garden is organized in 10-foot rows. The corn and tomatoes rows are 2 feet wide, and the beans rows are 3 feet wide. I like tomatoes the most and beans the least, and on a scale of 1 to 10, I would assign 10 to tomatoes, 7 to corn, and 3 to beans. Regardless of my preferences, my wife insists that I plant at least one row of green beans and no more than two rows of tomatoes. How many rows of each vegetable should I plant?

7. Habitat for Humanity is a wonderful charity organization that builds homes for needy families using volunteer labor. An eligible family can choose from three home sizes: 1000, 1100, and 1200 ft$^2$. Each size house requires a certain number of labor volunteers. The Fayetteville chapter has received five applications for the upcoming 6 months. The committee in charge assigns a score to each application based on several factors. A higher score signifies more need. For the next 6 months, the Fayetteville chapter can count on a maximum of 23 volunteers. The following data summarize the scores for the applications and the required number of volunteers. Which applications should the committee approve?

<table>
<thead>
<tr>
<th>Application</th>
<th>House size (ft$^2$)</th>
<th>Score</th>
<th>Required no. of volunteers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1200</td>
<td>78</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>64</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1100</td>
<td>68</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>62</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1200</td>
<td>85</td>
<td>8</td>
</tr>
</tbody>
</table>

8. Sheriff Bassam is up for reelection in Washington county. The funds available for the campaign are about $10,000. Although the reelection committee would like to launch the campaign in all five precincts of the county, limited funds dictate otherwise. The following table lists the voting population and the amount of funds needed to launch an effective campaign in each precinct. The choice for each precinct is to receive either all allotted funds or none. How should the funds be allocated?

<table>
<thead>
<tr>
<th>Precinct</th>
<th>Population</th>
<th>Required funds ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3100</td>
<td>3500</td>
</tr>
<tr>
<td>2</td>
<td>2600</td>
<td>2500</td>
</tr>
<tr>
<td>3</td>
<td>3500</td>
<td>4000</td>
</tr>
<tr>
<td>4</td>
<td>2800</td>
<td>3000</td>
</tr>
<tr>
<td>5</td>
<td>2400</td>
<td>2000</td>
</tr>
</tbody>
</table>

9. An electronic device consists of three components. The three components are in series so that the failure of one component causes the failure of the device. The reliability (probability of no failure) of the device can be improved by installing one or two standby units in each component. The following table charts the reliability, $r$, and the cost, $c$. The total capital available for the construction of the device is $10,000. How should the device be constructed? (Hint: The objective is to maximize the reliability, $r_1r_2r_3$, of the device. This means that the decomposition of the objective function is multiplicative rather than additive.)
10. Solve the following model by DP:

\[
\text{Maximize } z = \prod_{i=1}^{n} y_i
\]

subject to

\[
y_1 + y_2 + \cdots + y_n = c
\]

\[
y_j \geq 0, j = 1, 2, \ldots, n
\]

(Hint: This problem is similar to Problem 9, except that the variables, \(y_j\), are continuous.)

11. Solve the following problem by DP:

\[
\text{Minimize } z = y_1^2 + y_2^2 + \cdots + y_n^2
\]

subject to

\[
\prod_{i=1}^{n} y_i = c
\]

\[
y_i > 0, i = 1, 2, \ldots, n
\]

12. Solve the following problem by DP:

\[
\text{Maximize } z = (y_1 + 2)^2 + y_2 y_3 + (y_4 - 5)^2
\]

subject to

\[
y_1 + y_2 + y_3 + y_4 \leq 5
\]

\[
y_i \geq 0 \text{ and integer, } i = 1, 2, 3, 4
\]

13. Solve the following problem by DP:

\[
\text{Minimize } z = \max \{ f(y_1), f(y_2), \ldots, f(y_n) \}
\]

subject to

\[
y_1 + y_2 + \cdots + y_n = c
\]

\[
y_i \geq 0, i = 1, 2, \ldots, n
\]

Provide the solution for the special case of \(n = 3\), \(c = 10\), and \(f(y_1) = y_1 + 5\), \(f(y_2) = 5y_2 + 3\), and \(f(y_3) = y_3 - 2\).

10.3.2 Work-Force Size Model

In some construction projects, hiring and firing are exercised to maintain a labor force that meets the needs of the project. Given that the activities of hiring and firing both
incur additional costs, how should the labor force be maintained throughout the life of the project?

Let us assume that the project will be executed over the span of \( n \) weeks and that the minimum labor force required in week \( i \) is \( b_i \) laborers. Theoretically, we can use hiring and firing to keep the work-force in week \( i \) exactly equal to \( b_i \). Alternatively, it may be more economical to maintain a labor force larger than the minimum requirements through new hiring. This is the case we will consider here.

Given that \( x_i \) is the actual number of laborers employed in week \( i \), two costs can be incurred in week \( i \): \( C_1(x_i - b_i) \), the cost of maintaining an excess labor force \( x_i - b_i \), and \( C_2(x_i - x_{i-1}) \), the cost of hiring additional laborers, \( x_i - x_{i-1} \). It is assumed that no additional cost is incurred when employment is discontinued.

The elements of the DP model are defined as follows:

1. Stage \( i \) is represented by week \( i \), \( i = 1, 2, \ldots , n \).
2. The alternatives at stage \( i \) are \( x_i \), the number of laborers in week \( i \).
3. The state at stage \( i \) is represented by the number of laborers available at stage (week) \( i - 1, x_{i-1} \).

The DP recursive equation is given as

\[
f_i(x_{i-1}) = \min_{x_i \geq b_i} \{ C_1(x_i - b_i) + C_2(x_i - x_{i-1}) + f_{i+1}(x_i) \}, i = 1, 2, \ldots , n
\]

\[f_{n+1}(x_n) = 0\]

The computations start at stage \( n \) with \( x_n = b_n \) and terminate at stage 1.

**Example 10.3-2**

A construction contractor estimates that the size of the work force needed over the next 5 weeks to be 5, 7, 8, 4, and 6 workers, respectively. Excess labor kept on the force will cost $300 per worker per week, and new hiring in any week will incur a fixed cost of $400 plus $200 per worker per week.

The data of the problem are summarized as

\[
b_1 = 5, b_2 = 7, b_3 = 8, b_4 = 4, b_5 = 6
\]

\[
C_1(x_i - b_i) = 3(x_i - b_i), x_i > b_i, i = 1, 2, \ldots , 5
\]

\[
C_2(x_i - x_{i-1}) = 4 + 2(x_i - x_{i-1}), x_i > x_{i-1}, i = 1, 2, \ldots , 5
\]

Cost functions \( C_1 \) and \( C_2 \) are in hundreds of dollars.

**Stage 5** (\( b_5 = 6 \))

<table>
<thead>
<tr>
<th>( x_4 )</th>
<th>( x_5 = 6 )</th>
<th>( C(x_5 - 6) + C(x_5 - x_4) )</th>
<th>( f_5(x_4) )</th>
<th>( x_5^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3(0) + 4 + 2(2) = 8</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3(0) + 4 + 2(1) = 6</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3(0) + 0 = 0</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Stage 4 \( (b_4 = 4) \)

\[
C_1(x_4 - 4) + C_2(x_4 - x_3) + f_3(x_4) \quad \text{Optimum solution}
\]

<table>
<thead>
<tr>
<th>( x_2 )</th>
<th>( x_4 = 4 )</th>
<th>( x_4 = 5 )</th>
<th>( x_4 = 6 )</th>
<th>( f_4(x_4) )</th>
<th>( x_4^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>( 3(0) + 0 + 8 = 8 )</td>
<td>( 3(1) + 0 + 6 = 9 )</td>
<td>( 3(2) + 0 + 0 = 6 )</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Stage 3 \( (b_3 = 8) \)

\[
C_1(x_3 - 8) + C_2(x_3 - x_2) + f_4(x_3) \quad \text{Optimum solution}
\]

<table>
<thead>
<tr>
<th>( x_2 )</th>
<th>( x_3 = 8 )</th>
<th>( f_3(x_3) )</th>
<th>( x_3^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>( 3(0) + 4 + 2(1) + 6 = 12 )</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>( 3(0) + 0 + 6 = 6 )</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

Stage 2 \( (b_2 = 7) \)

\[
C_1(x_2 - 7) + C_2(x_2 - x_1) + f_3(x_2) \quad \text{Optimum solution}
\]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 = 7 )</th>
<th>( x_2 = 8 )</th>
<th>( f_3(x_2) )</th>
<th>( x_3^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( 3(0) + 4 + 2(2) + 12 = 20 )</td>
<td>( 3(1) + 4 + 2(3) + 6 = 19 )</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>( 3(0) + 4 + 2(1) + 12 = 18 )</td>
<td>( 3(1) + 4 + 2(2) + 6 = 17 )</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>( 3(0) + 0 + 12 = 12 )</td>
<td>( 3(1) + 4 + 2(1) + 6 = 15 )</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>( 3(0) + 0 + 12 = 12 )</td>
<td>( 3(1) + 0 + 6 = 9 )</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

Stage 1 \( (b_1 = 5) \)

\[
C_1(x_1 - 5) + C_2(x_1 - x_0) + f_1(x_1) \quad \text{Optimum solution}
\]

<table>
<thead>
<tr>
<th>( x_0 )</th>
<th>( x_2 = 7 )</th>
<th>( x_1 = 6 )</th>
<th>( x_1 = 7 )</th>
<th>( x_1 = 8 )</th>
<th>( f_1(x_0) )</th>
<th>( x_1^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( 3(0) + 4 + 2(5) + 19 = 33 )</td>
<td>( 3(1) + 4 + 2(6) + 17 = 36 )</td>
<td>( 3(2) + 4 + 2(7) + 12 = 36 )</td>
<td>( 3(2) + 4 + 2(8) + 9 = 35 )</td>
<td>33</td>
<td>5</td>
</tr>
</tbody>
</table>

The optimum solution is determined as

\[ x_0 = 0 \rightarrow x_1^* = 5 \rightarrow x_2^* = 8 \rightarrow x_3^* = 8 \rightarrow x_4^* = 6 \rightarrow x_5^* = 6 \]

The solution can be translated to the following plan:

<table>
<thead>
<tr>
<th>Week ( i )</th>
<th>Minimum labor force ( (b_i) )</th>
<th>Actual labor force ( (x_i) )</th>
<th>Decision</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>Hire 5 workers</td>
<td>( 4 + 2 \times 5 = 14 )</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>8</td>
<td>Hire 3 workers</td>
<td>( 4 + 2 \times 3 + 1 \times 3 = 13 )</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
<td>No change</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>Fire 2 workers</td>
<td>( 3 \times 2 = 6 )</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
<td>No change</td>
<td>0</td>
</tr>
</tbody>
</table>

The total cost is \( f_1(0) = \$3300 \).
PROBLEM SET 10.3B

1. Solve Example 10.3.2 for each of the following minimum labor requirements:
   * (a) \( b_1 = 6, b_2 = 5, b_3 = 3, b_4 = 6, b_5 = 8 \)
   (b) \( b_1 = 8, b_2 = 4, b_3 = 7, b_4 = 8, b_5 = 2 \)

2. In Example 10.3-2, if a severance pay of $100 is incurred for each fired worker, determine
   the optimum solution.

*3. Luxor Travel arranges 1-week tours to southern Egypt. The agency is contracted to pro-
   vide tourist groups with 7, 4, 7, and 8 rental cars over the next 4 weeks, respectively. Luxor
   Travel subcontracts with a local car dealer to supply rental needs. The dealer charges a
   rental fee of $220 per car per week, plus a flat fee of $500 for any rental transaction.
   Luxor, however, may elect not to return the rental cars at the end of the week, in which
   case the agency will be responsible only for the weekly rental ($220). What is the best way
   for Luxor Travel to handle the rental situation?

4. GECO is contracted for the next 4 years to supply aircraft engines at the rate of four engines
   a year. Available production capacity and production costs vary from year to year. GECO
   can produce five engines in year 1, six in year 2, three in year 3, and five in year 4. The corre-
   sponding production costs per engine over the next 4 years are $300,000, $330,000, $350,000,
   and $420,000, respectively. GECO can elect to produce more than it needs in a certain year,
   in which case the engines must be properly stored until shipment date. The storage cost per
   engine also varies from year to year, and is estimated to be $20,000 for year 1, $30,000 for
   year 2, $40,000 for year 3, and $50,000 for year 4. Currently, at the start of year 1, GECO has
   one engine ready for shipping. Develop an optimal production plan for GECO.

10.3.3 Equipment Replacement Model

The longer a machine stays in service, the higher is its maintenance cost, and the lower its
productivity. When a machine reaches a certain age, it may be more economical to replace
it. The problem thus reduces to determining the most economical age of a machine.

Suppose that we are studying the machine replacement problem over a span of \( n \)
years. At the start of each year, we decide whether to keep the machine in service an
extra year or to replace it with a new one. Let \( r(t), c(t), \) and \( s(t) \) represent the yearly
revenue, operating cost, and salvage value of a \( t \)-year-old machine. The cost of acquir-
ing a new machine in any year is \( I \).

The elements of the DP model are

1. **Stage** \( i \) is represented by year \( i, i = 1, 2, \ldots, n \).
2. The **alternatives** at stage (year) \( i \) call for either **keeping** or **replacing** the machine
   at the start of year \( i \).
3. The **state** at stage \( i \) is the age of the machine at the start of year \( i \).

Given that the machine is \( t \) years old at the start of year \( i \), define

\[
f_i(t) = \text{maximum net income for years } i, i + 1, \ldots, n
\]

The recursive equation is derived as

\[
f_i(t) = \begin{cases} 
  r(t) - c(t) + f_{i+1}(t+1), & \text{if KEEP} \\
  r(0) + s(t) - I - c(0) + f_{i+1}(1), & \text{if REPLACE} 
\end{cases}
\]

\[
f_{n+1}(\cdot) = 0
\]
Example 10.3-3

A company needs to determine the optimal replacement policy for a current 3-year-old machine over the next 4 years \((n = 4)\). The company requires that a 6-year-old machine be replaced. The cost of a new machine is $100,000. The following table gives the data of the problem.

<table>
<thead>
<tr>
<th>Age, (t) (yr)</th>
<th>Revenue, (r(t)) ($)</th>
<th>Operating cost, (c(t)) ($)</th>
<th>Salvage value, (s(t)) ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20,000</td>
<td>200</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>19,000</td>
<td>600</td>
<td>80,000</td>
</tr>
<tr>
<td>2</td>
<td>18,500</td>
<td>1200</td>
<td>60,000</td>
</tr>
<tr>
<td>3</td>
<td>17,200</td>
<td>1500</td>
<td>50,000</td>
</tr>
<tr>
<td>4</td>
<td>15,500</td>
<td>1700</td>
<td>30,000</td>
</tr>
<tr>
<td>5</td>
<td>14,000</td>
<td>1800</td>
<td>10,000</td>
</tr>
<tr>
<td>6</td>
<td>12,200</td>
<td>2200</td>
<td>5000</td>
</tr>
</tbody>
</table>

The determination of the feasible values for the age of the machine at each stage is somewhat tricky. Figure 10.6 summarizes the network representing the problem. At the start of year 1, we have a 3-year-old machine. We can either replace it \((R)\) or keep it \((K)\) for another year. At the start of year 2, if replacement occurs, the new machine will be 1 year old; otherwise, the old machine will be 4 years old. The same logic applies at the start of years 2 to 4. If a 1-year-old machine is replaced at the start of year 2, 3, or 4, its replacement will be 1 year old at the start of the following year. Also, at the start of year 4, a 6-year-old machine must be replaced, and at the end of year 4 (end of the planning horizon), we salvage \((S)\) the machines.

FIGURE 10.6

Representation of machine age as a function of decision year in Example 10.3-3

\[ K = \text{Keep} \]
\[ R = \text{Replace} \]
\[ S = \text{Salvage} \]
The network shows that at the start of year 2, the possible ages of the machine are 1 and 4 years. For the start of year 3, the possible ages are 1, 2, and 5 years, and for the start of year 4, the possible ages are 1, 2, 3, and 6 years.

The solution of the network in Figure 10.6 is equivalent to finding the longest route (i.e., maximum revenue) from the start of year 1 to the end of year 4. We will use the tabular form to solve the problem. All values are in thousands of dollars. Note that if a machine is replaced in year 4 (i.e., end of the planning horizon), its revenue will include the salvage value, $s(t)$, of the replaced machine and the salvage value, $s(1)$, of the replacement machine.

### Stage 4

<table>
<thead>
<tr>
<th>$t$</th>
<th>$K$</th>
<th>$R$</th>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.0 + 60 - 0.6 = 78.4</td>
<td>20 + 80 + 80 - 0.2 - 100 = 79.8</td>
<td>79.8</td>
</tr>
<tr>
<td>2</td>
<td>18.5 + 50 - 1.2 = 67.3</td>
<td>20 + 60 + 80 - 0.2 - 100 = 79.8</td>
<td>67.3</td>
</tr>
<tr>
<td>3</td>
<td>17.2 + 30 - 1.5 = 45.7</td>
<td>20 + 50 + 80 - 0.2 - 100 = 49.8</td>
<td>49.8</td>
</tr>
<tr>
<td>6</td>
<td>(Must replace)</td>
<td>20 + 5 + 80 - 0.2 - 100 = 4.8</td>
<td>4.8</td>
</tr>
</tbody>
</table>

### Stage 3

<table>
<thead>
<tr>
<th>$t$</th>
<th>$K$</th>
<th>$R$</th>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.0 - 0.6 + 67.3 = 85.7</td>
<td>20 + 80 - 0.2 - 100 + 79.8 = 85.6</td>
<td>85.7</td>
</tr>
<tr>
<td>2</td>
<td>18.5 - 1.2 + 49.8 = 67.1</td>
<td>20 + 60 - 0.2 - 100 + 79.8 = 59.6</td>
<td>67.1</td>
</tr>
<tr>
<td>3</td>
<td>14.0 - 1.8 + 4.8 = 17.0</td>
<td>20 + 10 - 0.2 - 100 + 79.8 = 19.6</td>
<td>19.6</td>
</tr>
</tbody>
</table>

### Stage 2

<table>
<thead>
<tr>
<th>$t$</th>
<th>$K$</th>
<th>$R$</th>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.0 - 0.6 + 67.1 = 85.5</td>
<td>20 + 80 - 0.2 - 100 + 85.7 = 85.5</td>
<td>85.5</td>
</tr>
<tr>
<td>4</td>
<td>15.5 - 1.7 + 19.6 = 33.4</td>
<td>20 + 30 - 0.2 - 100 + 85.7 = 35.5</td>
<td>35.5</td>
</tr>
</tbody>
</table>

### Stage 1

<table>
<thead>
<tr>
<th>$t$</th>
<th>$K$</th>
<th>$R$</th>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>17.2 - 1.5 + 35.5 = 51.2</td>
<td>20 + 50 - 0.2 - 100 + 85.5 = 55.3</td>
<td>55.3</td>
</tr>
</tbody>
</table>
Figure 10.7 summarizes the optimal solution. At the start of year 1, given $t = 3$, the optimal decision is to replace the machine. Thus, the new machine will be 1 year old at the start of year 2, and $t = 1$ at the start of year 2 calls for either keeping or replacing the machine. If it is replaced, the new machine will be 1 year old at the start of year 3; otherwise, the kept machine will be 2 years old. The process is continued in this manner until year 4 is reached.

The alternative optimal policies starting in year 1 are $(R, K, K, R)$ and $(R, R, K, K)$. The total cost is $55,300.

**PROBLEM SET 10.3C**

1. In each of the following cases, develop the network, and find the optimal solution for the model in Example 10.3-3:
   (a) The machine is 2 years old at the start of year 1.
   (b) The machine is 1 year old at the start of year 1.
   (c) The machine is bought new at the start of year 1.

2. My son, age 13, has a lawn-mowing business with 10 customers. For each customer, he cuts the grass 3 times a year, which earns him $50 for each mowing. He has just paid $200 for a new mower. The maintenance and operating cost of the mower is $120 for the first year in service, and increases by 20% a year thereafter. A 1-year-old mower has a resale value of $150, which decreases by 10% a year thereafter. My son, who plans to keep his business until he is 16, thinks that it is more economical to buy a new mower every 2 years. He bases his decision on the fact that the price of a new mower will increase only by 10% a year. Is his decision justified?

3. Circle Farms wants to develop a replacement policy for its 2-year-old tractor over the next 5 years. A tractor must be kept in service for at least 3 years, but must be disposed of after 5 years. The current purchase price of a tractor is $40,000 and increases by 10% a year. The salvage value of a 1-year-old tractor is $30,000 and decreases by 10% a year. The current annual operating cost of the tractor is $1300 but is expected to increase by 10% a year.
   (a) Formulate the problem as a shortest-route problem.
   (b) Develop the associated recursive equation.
   (c) Determine the optimal replacement policy of the tractor over the next 5 years.

4. Consider the equipment replacement problem over a period of $n$ years. A new piece of equipment costs $c$ dollars, and its resale value after $t$ years in operation is $s(t) = n - t$
for \( n > t \) and zero otherwise. The annual revenue is a function of the age \( t \) and is given by
\[
r(t) = n^2 - t^2 \quad \text{for} \quad n > t \quad \text{and zero otherwise}.
\]

(a) Formulate the problem as a DP model.

(b) Find the optimal replacement policy given that \( c = \$10,000 \), \( n = 5 \), and the equipment is 2 years old.

5. Solve Problem 4, assuming that the equipment is 1 year old and that \( n = 4 \), \( c = \$6000 \),
\[
r(t) = \frac{2}{t + 1}.
\]

10.3.4 Investment Model

Suppose that you want to invest the amounts \( P_1, P_2, \ldots, P_n \) at the start of each of the next \( n \) years. You have two investment opportunities in two banks: First Bank pays an interest rate \( r_1 \) and Second Bank pays \( r_2 \), both compounded annually. To encourage deposits, both banks pay bonuses on new investments in the form of a percentage of the amount invested. The respective bonus percentages for First Bank and Second Bank are \( q_{1i} \) and \( q_{2i} \) for year \( i \). Bonuses are paid at the end of the year in which the investment is made and may be reinvested in either bank in the immediately succeeding year. This means that only bonuses and fresh new money may be invested in either bank. However, once an investment is deposited, it must remain in the bank until the end of the \( n \)-year horizon. Devise the investment schedule over the next \( n \) years.

The elements of the DP model are

1. **Stage** \( i \) is represented by year \( i, i = 1, 2, \ldots, n \).

2. The **alternatives** at stage \( i \) are \( I_i \) and \( \bar{I}_i \), the amounts invested in First Bank and Second Bank, respectively.

3. The **state**, \( x_i \), at stage \( i \) is the amount of capital available for investment at the start of year \( i \).

We note that \( \bar{I}_i = x_i - I_i \) by definition. Thus
\[
x_1 = P_1
\]
\[
x_i = P_i + q_{i-1,1}I_{i-1} + q_{i-1,2}(x_{i-1} - I_{i-1})
\]
\[
= P_i + (q_{i-1,1} - q_{i-1,2})I_{i-1} + q_{i-1,2}x_{i-1}, \quad i = 2, 3, \ldots, n
\]

The reinvestment amount \( x_i \) includes only new money plus any bonus from investments made in year \( i - 1 \).

Define
\[
f_i(x_i) = \text{optimal value of the investments for years} \ i, i+1, \ldots, \text{and} \ n, \text{given} \ x_i
\]

Next, define \( s_i \) as the accumulated sum at the end of year \( n \), given that \( I_i \) and \( (x_i - I_i) \) are the investments made in year \( i \) in First Bank and Second Bank, respectively. Letting \( \alpha_k = (1 + r_k), k = 1, 2 \), the problem can be stated as

Maximize \( z = s_1 + s_2 + \cdots + s_n \)

where
\[
s_i = I_i\alpha_{i+1-i} + (x_i - I_i)\alpha_{i+1-i}
\]
\[
= (\alpha_1^{i+1-i} - \alpha_2^{i+1-i})I_i + \alpha_2^{i+1-i}x_i, \quad i = 1, 2, \ldots, n - 1
\]
\[
s_n = (\alpha_1 + q_{n1} - \alpha_2 - q_{n2})I_n + (\alpha_2 + q_{n2})x_n
\]
The terms $q_{n1}$ and $q_{n2}$ in $s_n$ are added because the bonuses for year $n$ are part of the final accumulated sum of money from the investment.

The backward DP recursive equation is thus given as

$$f_i(x_i) = \max_{0 \leq s_i \leq x_i} \{ s_i + f_{i+1}(x_{i+1}) \}, \quad i = 1, 2, \ldots, n - 1$$

$$f_{n+1}(x_{n+1}) = 0$$

As given previously, $x_{i+1}$ is defined in terms of $x_i$.

Example 10.3-4

Suppose that you want to invest $4000 now and $2000 at the start of years 2 to 4. The interest rate offered by First Bank is 8% compounded annually, and the bonuses over the next 4 years are 1.8%, 1.7%, 2.1%, and 2.5%, respectively. The annual interest rate offered by Second Bank is .2% lower than that of First Bank, but its bonus is .5% higher. The objective is to maximize the accumulated capital at the end of 4 years.

Using the notation introduced previously, we have

$$P_1 = 4000, P_2 = P_3 = P_4 = 2000$$

$$\alpha_1 = (1 + .08) = 1.08$$

$$\alpha_2 = (1 + .078) = 1.078$$

$$q_{11} = .018, q_{21} = .017, q_{31} = .021, q_{41} = .025$$

$$q_{12} = .023, q_{22} = .022, q_{32} = .026, q_{42} = .030$$

Stage 4

$$f_4(x_4) = \max_{0 \leq I_4 \leq x_4} \{ s_4 \}$$

where

$$s_4 = (\alpha_1 + q_{41} - \alpha_2 - q_{42})I_4 + (\alpha_2 + q_{42})x_4 = -.003I_4 + 1.108x_4$$

The function $s_4$ is linear in $I_4$ in the range $0 \leq I_4 \leq x_4$ and its maximum occurs at $I_4 = 0$ because of the negative coefficient of $I_4$. Thus, the optimum solution for stage 5 can be summarized as

<table>
<thead>
<tr>
<th>State</th>
<th>$f_4(x_4)$</th>
<th>$I_4^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_4$</td>
<td>1.108$x_4$</td>
<td>0</td>
</tr>
</tbody>
</table>

Stage 3

$$f_3(x_3) = \max_{0 \leq I_3 \leq x_3} \{ s_3 + f_4(x_4) \}$$

where

$$s_3 = (1.08^3 - 1.078^3)I_3 + 1.078^3x_3 = .00432I_3 + 1.1621x_3$$

$$x_4 = 2000 - .005I_3 + .026x_3$$
Thus,
\[
f_3(x_3) = \max_{0 \leq I_3 \leq x_3} \{0.00432I_3 + 1.1621x_3 + 1.108(2000 - .005I_3 + 0.026x_3)\}
\]
\[
= \max_{0 \leq I_3 \leq x_3} \{2216 - .00122I_3 + 1.1909x_3\}
\]

<table>
<thead>
<tr>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
</tr>
<tr>
<td>( x_3 )</td>
</tr>
</tbody>
</table>

Stage 2

\[
f_2(x_2) = \max_{0 \leq I_2 \leq x_2} \{s_2 + f_3(x_3)\}
\]

where
\[
s_2 = (1.08^3 - 1.078^3)I_2 + 1.078^3x_2 = .006985I_2 + 1.25273x_2
\]
\[
x_3 = 2000 - .005I_2 + .022x_2
\]

Thus,
\[
f_2(x_2) = \max_{0 \leq I_2 \leq x_2} \{0.006985I_2 + 1.25273x_2 + 2216 + 1.1909(2000 - .005I_2 + .022x_2)\}
\]
\[
= \max_{0 \leq I_2 \leq x_2} \{4597.8 + .0010305I_2 + 1.27893x_2\}
\]

<table>
<thead>
<tr>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
</tr>
<tr>
<td>( x_2 )</td>
</tr>
</tbody>
</table>

Stage 1

\[
f_1(x_1) = \max_{0 \leq I_1 \leq x_1} \{s_1 + f_2(x_2)\}
\]

where
\[
s_1 = (1.08^4 - 1.078^4)I_1 + 1.078^4x_1 = .01005I_1 + 1.3504x_1
\]
\[
x_2 = 2000 - .005I_1 + .023x_1
\]

Thus,
\[
f_1(x_1) = \max_{0 \leq I_1 \leq x_1} \{.01005I_1 + 1.3504x_1 + 4597.8 + 1.27996(2000 - .005I_1 + .023x_1)\}
\]
\[
= \max_{0 \leq I_1 \leq x_1} \{7157.7 + .00365I_1 + 1.37984x_1\}
\]

<table>
<thead>
<tr>
<th>Optimum solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
</tr>
<tr>
<td>( x_1 )</td>
</tr>
</tbody>
</table>
Working backward and noting that $I_1^* = 4000$, $I_2^* = x_2$, $I_3^* = I_4^* = 0$, we get

\[
x_1 = 4000
\]
\[
x_2 = 2000 - .005 \times 4000 + .023 \times 4000 = \$2072
\]
\[
x_3 = 2000 - .005 \times 2072 + .022 \times 2072 = \$2035.22
\]
\[
x_4 = 2000 - .005 \times 0 + .026 \times \$2035.22 = \$2052.92
\]

The optimum solution is thus summarized as

<table>
<thead>
<tr>
<th>Year</th>
<th>Optimum solution</th>
<th>Decision</th>
<th>Accumulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$I_1^* = x_1$</td>
<td>Invest $x_1 = $4000$ in First Bank</td>
<td>$s_1 = $5441.80$</td>
</tr>
<tr>
<td>2</td>
<td>$I_2^* = x_2$</td>
<td>Invest $x_2 = $2072$ in First Bank</td>
<td>$s_2 = $2610.13$</td>
</tr>
<tr>
<td>3</td>
<td>$I_3^* = 0$</td>
<td>Invest $x_3 = $2035.22$ in Second Bank</td>
<td>$s_3 = $2365.13$</td>
</tr>
<tr>
<td>4</td>
<td>$I_4^* = 0$</td>
<td>Invest $x_4 = $2052.92$ in Second Bank</td>
<td>$s_4 = $2274.64$</td>
</tr>
</tbody>
</table>

Total accumulation = $f_4(x_1) = 7157.7 + 1.38349(4000) = \$12,691.66 \approx s_1 + s_2 + s_3 + s_4$

**PROBLEM SET 10.3D**

1. Solve Example 10.3-4, assuming that $r_1 = .085$ and $r_2 = .08$. Additionally, assume that $P_1 = \$5000$, $P_2 = \$4000$, $P_3 = \$3000$, and $P_4 = \$2000$.

2. An investor with an initial capital of $\$10,000$ must decide at the end of each year how much to spend and how much to invest in a savings account. Each dollar invested returns $\alpha = \$1.09$ at the end of the year. The satisfaction derived from spending $\$y$ in any one year is quantified by the equivalence of owning $\$\sqrt{y}$. Solve the problem by DP for a span of 5 years.

3. A farmer owns $k$ sheep. At the end of each year, a decision is made as to how many to sell or keep. The profit from selling a sheep in year $i$ is $P_i$. The sheep kept in year $i$ will double in number in year $i + 1$. The farmer plans to sell out completely at the end of $n$ years.

   *(a)* Derive the general recursive equation for the problem.

   *(b)* Solve the problem for $n = 3$ years, $k = 2$ sheep, $P_1 = \$100$, $P_2 = \$130$, and $P_3 = \$120$.

**10.3.5 Inventory Models**

DP has important applications in the area of inventory control. Chapters 11 and 14 present some of these applications. The models in Chapter 11 are deterministic, and those in Chapter 14 are probabilistic.
10.4 PROBLEM OF DIMENSIONALITY

In all the DP models we presented, the state at any stage is represented by a single element. For example, in the knapsack model (Section 10.3.1), the only restriction is the weight of the item. More realistically, the volume of the knapsack may also be another viable restriction. In such a case, the state at any stage is said to be two-dimensional because it consists of two elements: weight and volume.

The increase in the number of state variables increases the computations at each stage. This is particularly clear in DP tabular computations because the number of rows in each tableau corresponds to all possible combinations of state variables. This computational difficulty is sometimes referred to in the literature as the **curse of dimensionality**.

The following example is chosen to demonstrate the problem of dimensionality. It also serves to show the relationship between linear and dynamic programming.

---

**Example 10.4-1**

Acme Manufacturing produces two products. The daily capacity of the manufacturing process is 430 minutes. Product 1 requires 2 minutes per unit, and product 2 requires 1 minute per unit. There is no limit on the amount produced of product 1, but the maximum daily demand for product 2 is 230 units. The unit profit of product 1 is $2 and that of product 2 is $5. Find the optimal solution by DP.

The problem is represented by the following linear program:

Maximize \( z = 2x_1 + 5x_2 \)

subject to

\[
\begin{align*}
2x_1 + x_2 & \leq 430 \\
x_2 & \leq 230 \\
x_1, x_2 & \geq 0
\end{align*}
\]

The elements of the DP model are

1. **Stage** \( i \) corresponds to product \( i, i = 1, 2. \)
2. **Alternative** \( x_i \) is the amount of product \( i, i = 1, 2. \)
3. **State** \( (v_2, w_2) \) represents the amounts of resources 1 and 2 (production time and demand limits) used in stage 2.
4. **State** \( (v_1, w_1) \) represents the amounts of resources 1 and 2 (production time and demand limits) used in stages 1 and 2.

**Stage 2.** Define \( f_2(v_2, w_2) \) as the maximum profit for stage 2 (product 2), given the state \( (v_2, w_2) \). Then

\[
f_2(v_2, w_2) = \max_{0 \leq x_2 \leq w_2} \{ 5x_2 \}
\]

Thus, \( \max \{ 5x_2 \} \) occurs at \( x_2 = \min \{ v_2, w_2 \} \), and the solution for stage 2 is
### 10.4 Problem of Dimensionality

**Optimum solution**

<table>
<thead>
<tr>
<th>State</th>
<th>( f_2(v_2, w_2) )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((v_2, w_2))</td>
<td>5 ( \min{v_2, w_2} )</td>
<td>( \min{v_2, w_2} )</td>
</tr>
</tbody>
</table>

**Stage 1**

\[
 f_1(v_1, w_1) = \max_{0 \leq 2x_1 \leq v_1} \{2x_1 + f_2(v_1 - 2x_1, w_1)\}
\]

\[
 f_1(v_1, w_1) = \max_{0 \leq 2x_1 \leq v_1} \{2x_1 + 5 \min(v_1 - 2x_1, w_1)\}
\]

The optimization of stage 1 requires the solution of a (generally difficult) minimax problem. For the present problem, we set \( v_1 = 430 \) and \( w_1 = 230 \), which gives \( 0 \leq 2x_1 \leq 430 \). Because \( \min(430 - 2x_1, 230) \) is the lower envelope of two intersecting lines (verify!), it follows that

\[
 \min(430 - 2x_1, 230) = \begin{cases} 
230, & 0 \leq x_1 \leq 100 \\
430 - 2x_1, & 100 \leq x_1 \leq 215 
\end{cases}
\]

and

\[
 f_1(430, 230) = \max_{0 \leq x_1 \leq 215} \{2x_1 + 5 \min(430 - 2x_1, 230)\}
\]

\[
 f_1(430, 230) = \max_{x_1} \begin{cases} 
2x_1 + 1150, & 0 \leq x_1 \leq 100 \\
-8x_1 + 2150, & 100 \leq x_1 \leq 215 
\end{cases}
\]

You can verify graphically that the optimum value of \( f_1(430, 230) \) occurs at \( x_1 = 100 \). Thus, we get

<table>
<thead>
<tr>
<th>State</th>
<th>( f_1(v_1, w_1) )</th>
<th>( x_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((430, 230))</td>
<td>1350</td>
<td>100</td>
</tr>
</tbody>
</table>

To determine the optimum value of \( x_2 \), we note that

\[
 v_2 = v_1 - 2x_1 = 430 - 200 = 230
\]

\[
 w_2 = w_1 - 0 = 230
\]

Consequently,

\[
 x_2 = \min\{v_2, w_2\} = 230
\]

The complete optimum solution is thus summarized as

\[
 x_1 = 100 \text{ units}, x_2 = 230 \text{ units}, z = $1350
\]
PROBLEM SET 10.4A

1. Solve the following problems by DP.
   (a) Maximize $z = 4x_1 + 14x_2$
       subject to
       \[
       \begin{align*}
       2x_1 + 7x_2 &\leq 21 \\
       7x_1 + 2x_2 &\leq 21 \\
       x_1, x_2 &\geq 0
       \end{align*}
       \]
   (b) Maximize $z = 8x_1 + 7x_2$
       subject to
       \[
       \begin{align*}
       2x_1 + x_2 &\leq 8 \\
       5x_1 + 2x_2 &\leq 15 \\
       x_1, x_2 &\geq 0 \text{ and integer}
       \end{align*}
       \]
   (c) Maximize $z = 7x_1^2 + 6x_1 + 5x_2^2$
       subject to
       \[
       \begin{align*}
       x_1 + 2x_2 &\leq 10 \\
       x_1 - 3x_2 &\leq 9 \\
       x_1, x_2 &\geq 0
       \end{align*}
       \]

2. In the $n$-item knapsack problem of Example 10.3-1, suppose that the weight and volume limitations are $W$ and $V$, respectively. Given that $w_i$, $v_i$, and $r_i$ are the weight, value, and revenue per unit of item $i$, write the DP backward recursive equation for the problem.

REFERENCES


