2. For the upcoming planting season, Farmer McCoy can plant corn (a_1), plant wheat (a_2), plant soybeans (a_3), or use the land for grazing (a_4). The payoffs associated with the different actions are influenced by the amount of rain: heavy rainfall (s_1), moderate rainfall (s_2), light rainfall (s_3), or drought season (s_4).

   The payoff matrix (in thousands of dollars) is estimated as

	s_1	s_2	s_3	s_4
a_1	−20	60	30	−5
a_2	40	50	35	0
a_3	−50	100	45	−10
a_4	12	15	15	10

   Develop a course of action for Farmer McCoy.

3. One of N machines must be selected for manufacturing Q units of a specific product. The minimum and maximum demands for the product are Q^* and Q^{**}, respectively. The total production cost for Q items on machine i involves a fixed cost K_i and a variable cost per unit c_i, and is given as

$$TC_i = K_i + c_iQ$$

   (a) Devise a solution for the problem under each of the four criteria of decisions under uncertainty.

   (b) For $1000 \le Q \le 4000$, solve the problem for the following set of data:

Machine i	K_i (\$)	C_i (\$)
1	100	5
2	40	12
3	150	3
4	90	8

13.4 GAME THEORY

Game theory deals with decision situations in which two *intelligent* opponents with conflicting objectives are trying to outdo one another. Typical examples include launching advertising campaigns for competing products and planning strategies for warring armies.

In a game conflict, two opponents, known as **players**, will each have a (finite or infinite) number of alternatives or **strategies**. Associated with each pair of strategies is a **payoff** that one player receives from the other. Such games are known as **two-person zero-sum games** because a gain by one player signifies an equal loss to the other. It suffices, then, to summarize the game in terms of the payoff to one player. Designating the two players as A and B with m and n strategies, respectively, the game is usually represented by the payoff matrix to player A as

	B_1	B_2	...	B_n
A_1	a_{11}	a_{12}	...	a_{1n}
A_2	a_{21}	a_{22}	...	a_{2n}
⋮	⋮	⋮	⋮	⋮
A_m	a_{m1}	a_{m2}	...	a_{mn}

The representation indicates that if A uses strategy i and B uses strategy j, the payoff to A is a_{ij}, which means that the payoff to B is $-a_{ij}$.

Real-Life Application—Ordering Golfers on the Final Day of Ryder Cup Matches

In the final day of a golf tournament, two teams compete for the championship. Each team captain must submit an ordered list of golfers (a *slate*) that automatically determines the matches. It is plausible to assume that if two competing players occupy the same order in their respective slates then there is 50-50 chance that either golfer will win the match. This probability will increase when a higher-order golfer is matched with a lower-order one. The goal is to develop an analytical procedure that will support or refute the idea of using slates. Case 12, Chapter 24 on the CD provides details on the study.

13.4.1 Optimal Solution of Two-Person Zero-Sum Games

Because games are rooted in conflict of interest, the optimal solution selects one or more strategies for each player such that any change in the chosen strategies does not improve the payoff to either player. These solutions can be in the form of a single pure strategy or several strategies mixed according to specific probabilities. The following two examples demonstrate the two cases.

Example 13.4-1

Two companies, A and B, sell two brands of flu medicine. Company A advertises in radio (A_1), television (A_2), and newspapers (A_3). Company B, in addition to using radio (B_1), television (B_2), and newspapers (B_3), also mails brochures (B_4). Depending on the effectiveness of each advertising campaign, one company can capture a portion of the market from the other. The following matrix summarizes the percentage of the market captured or lost by company A.

	B_1	B_2	B_3	B_4	Row min
A_1	8	−2	9	−3	−3
A_2	6	**5**	6	8	**5** ← Maximin
A_3	−2	4	−9	5	−9
Column max	8	**5**	9	8	
		↑ Minimax			

Find the value of the (Game)

The solution of the game is based on the principle of securing the *best of the worst* for each player. If Company A selects strategy A_1, then regardless of what B does, the worst that can happen is that A loses 3% of the market share to B. This is represented by the minimum value of the entries in row 1. Similarly, the strategy A_2 worst outcome is for A to capture 5% of the market from B, and the strategy A_3 worst outcome is for A to lose 9% to B. These result are listed in the "row min" column. To achieve the *best* of the *worst*, Company A chooses strategy A_2 because it corresponds to the maximin value, or the largest element in the "row min" column.

Next, consider Company B's strategy. Because the given payoff matrix is for A, *B's best of the worst* criterion requires determining the minimax value. The result is that Company B should select strategy B_2.

The optimal solution of the game calls for selecting strategies A_2 and B_2, which means that both companies should use television advertising. The payoff will be in favor of company A, because its market share will increase by 5%. In this case, we say that the **value of the game** is 5%, and that A and B are using a **saddle-point** solution.

The saddle-point solution precludes the selection of a better strategy by either company. If B moves to another strategy (B_1, B_3, or B_4), Company A can stay with strategy A_2, which ensures that B will lose a worse share of the market (6% or 8%). By the same token, A does not want to use a different strategy because if A moves to strategy A_3, B can move to B_3 and realize a 9% increase in market share. A similar conclusion is realized if A moves to A_1, as B can move to B_4 and realize a 3% increase in market share.

The optimal saddle-point solution of a game need not be a pure strategy. Instead, the solution may require mixing two or more strategies randomly, as the following example illustrates.

Example 13.4-2

Two players, A and B, play the coin-tossing game. Each player, unbeknownst to the other, chooses a head (H) or a tail (T). Both players would reveal their choices simultaneously. If they match (HH or TT), player A receives \$1 from B. Otherwise, A pays B \$1.

The following payoff matrix for player A gives the row-min and the column-max values corresponding to A's and B's strategies, respectively.

	B_H	B_T	Row min
A_H	1	−1	−1
A_T	−1	1	−1
Column max	1	1	

The maximin and the minimax values of the games are −\$1 and \$1, respectively. Because the two values are not equal, the game does not have a pure strategy solution. In particular, if A_H is used by player A, player B will select B_T to receive \$1 from A. If this happens, A can move to strategy A_T to reverse the outcome of the game by receiving \$1 from B. The constant temptation to switch to another strategy shows that a pure strategy solution is not acceptable. Instead, both players can randomly mix their respective pure strategies. In this case, the optimal value of the game will occur somewhere between the maximin and the minimax values of the game—that is,

$$\text{maximin (lower) value} \le \text{value of the game} \le \text{minimax (upper) value}$$

(See Problem 5, Set 13.4a.) Thus, in the coin-tossing example, the value of the game must lie between $-\$1$ and $+\$1$.

PROBLEM SET 13.4A

1. Determine the saddle-point solution, the associated pure strategies, and the value of the game for each of the following games. The payoffs are for player A.

***(a)**

	B_1	B_2	B_3	B_4
A_1	8	6	2	8
A_2	8	9	4	5
A_3	7	5	3	5

(b)

	B_1	B_2	B_3	B_4
A_1	4	−4	−5	6
A_2	−3	−4	−9	−2
A_3	6	7	−8	−9
A_4	7	3	−9	5

2. The following games give A's payoff. Determine the values of p and q that will make the entry (2, 2) of each game a saddle point:

(a)

	B_1	B_2	B_3
A_1	1	q	6
A_2	p	5	10
A_3	6	2	3

(b)

	B_1	B_2	B_3
A_1	2	4	5
A_2	10	7	q
A_3	4	p	6

3. Specify the range for the value of the game in each of the following cases, assuming that the payoff is for player A:

***(a)**

	B_1	B_2	B_3	B_4
A_1	1	9	6	0
A_2	2	3	8	4
A_3	−5	−2	10	−3
A_4	7	4	−2	−5

(b)

	B_1	B_2	B_3	B_4
A_1	−1	9	6	8
A_2	−2	10	4	6
A_3	5	3	0	7
A_4	7	−2	8	4

(c)

	B_1	B_2	B_3
A_1	3	6	1
A_2	5	2	3
A_3	4	2	−5

(d)

	B_1	B_2	B_3	B_4
A_1	3	7	1	3
A_2	4	8	0	−6
A_3	6	−9	−2	4

4. Two companies promote two competing products. Currently, each product controls 50% of the market. Because of recent improvements in the two products, each company is preparing to launch an advertising campaign. If neither company advertises, equal market

shares will continue. If either company launches a stronger campaign, the other is certain to lose a proportional percentage of its customers. A survey of the market shows that 50% of potential customers can be reached through television, 30% through newspapers, and 20% through radio.

(a) Formulate the problem as a two-person zero-sum game, and select the appropriate advertising media for each company.

(b) Determine a range for the value of the game. Can each company operate with a single pure strategy?

5. Let a_{ij} be the (i, j)th element of a payoff matrix with m strategies for player A and n strategies for player B. The payoff is for player A. Prove that

$$\max_i \min_j a_{ij} \le \min_j \max_i a_{ij}$$

13.4.2 Solution of Mixed Strategy Games

Games with mixed strategies can be solved either graphically or by linear programming. The graphical solution is suitable for games in which at least one player has exactly two pure strategies. The method is interesting because it explains the idea of a saddle point graphically. Linear programming can be used to solve any two-person zero-sum game.

Graphical Solution of Games. We start with the case of $(2 \times n)$ games in which player A has two strategies.

		y_1	y_2	$\cdots$	y_n
		B_1	B_2	$\cdots$	B_n
x_1:	A_1	a_{11}	a_{12}	$\cdots$	a_{1m}
$1 - x_1$:	A_2	a_{21}	a_{22}	$\cdots$	a_{2m}

The game assumes that player A mixes strategies A_1 and A_2 with the respective probabilities x_1 and $1 - x_1$, $0 \le x_1 \le 1$. Player B mixes strategies B_1 to B_n with the probabilities $y_1, y_2, \ldots$, and y_n, where $y_j \ge 0$ for $j = 1, 2, \ldots, n$, and $y_1 + y_2 + \cdots + y_n = 1$. In this case, A's expected payoff corresponding to B's jth pure strategy is computed as

$$(a_{1j} - a_{2j})x_1 + a_{2j}, j = 1, 2, \ldots, n$$

Player A thus seeks to determine the value of x_1 that maximizes the minimum expected payoffs—that is,

$$\max_{x_1} \min_j \{(a_{1j} - a_{2j})x_1 + a_{2j}\}$$

Example 13.4-3

Consider the following 2×4 game. The payoff is for player A.

	B_1	B_2	B_3	B_4
A_1	2	2	3	−1
A_2	4	3	2	6

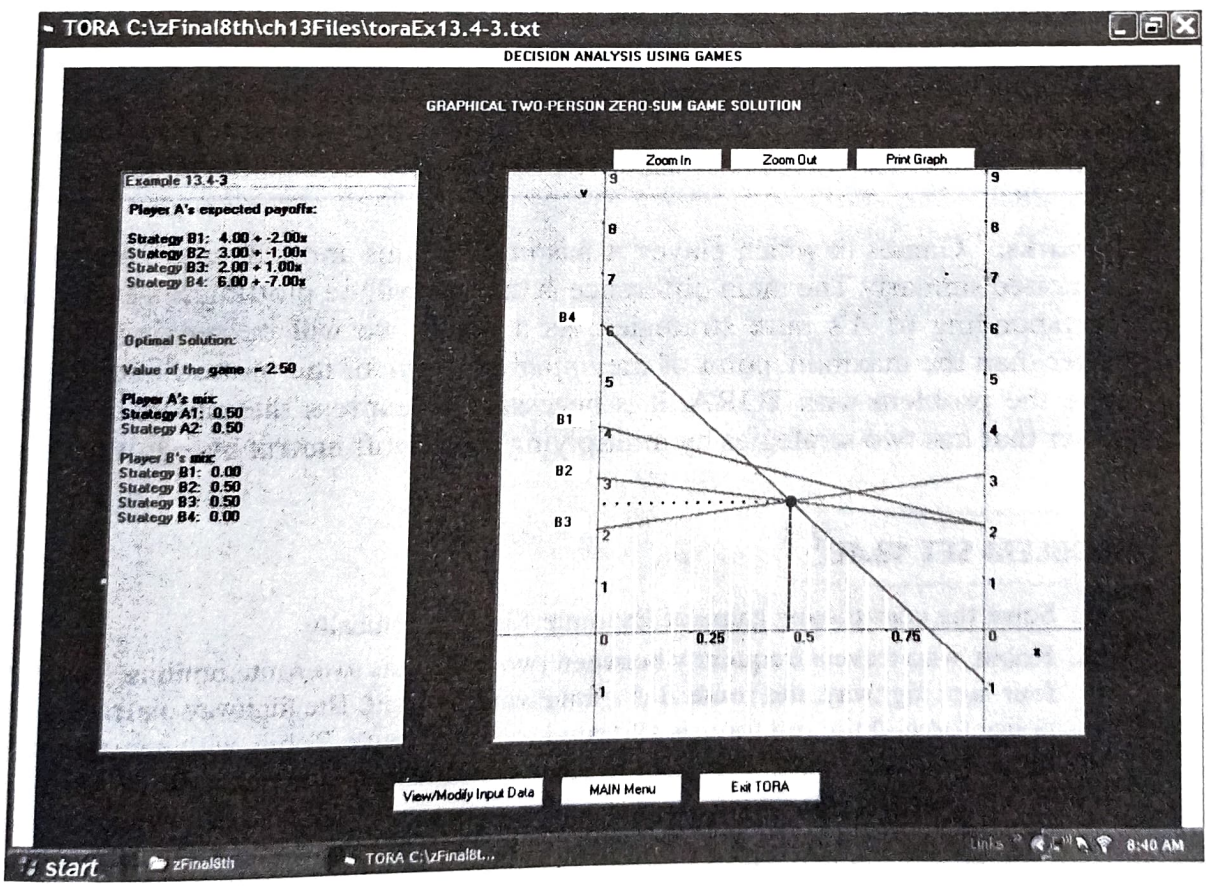
The game has no pure strategy solution. A's expected payoffs corresponding to B's pure strategies are given as

B's pure strategy	A's expected payoff
1	$-2x_1 + 4$
2	$-x_1 + 3$
3	$x_1 + 2$
4	$-7x_1 + 6$

Figure 13.9 provides TORA plot of the four straight lines associated with B's pure strategies (file toraEx13.4-3.txt).[3] To determine the *best of the worst*, the lower envelope of the four lines (delineated by vertical stripes) represents the minimum (worst) expected payoff for A regardless of what B does. The maximum (best) of the lower envelope corresponds to the maximin solution point at $x_1^* = .5$. This point is the intersection of lines associated with strategies B_3 and B_4. Player A's optimal solution thus calls for mixing A_1 and A_2 with probabilities .5, and .5, respectively.

FIGURE 13.9

TORA graphical solution of the two-person zero-sum game of Example 13.4-3 (file toraEx13.4-3.txt)



[3]From Main menu select Zero-sum Games and input the problem data, then select Graphical from the SOLVE/MODIFY menu.

The corresponding value of the game, v, is determined by substituting $x_1 = .5$ in either of the functions for lines 3 and 4, which gives

$$v = \begin{cases} \frac{1}{2} + 2 = \frac{5}{2}, & \text{from line 3} \\ -7\left(\frac{1}{2}\right) + 6 = \frac{5}{2}, & \text{from line 4} \end{cases}$$

Player B's optimal mix is determined by the two strategies that define the lower envelope of the graph. This means that B can mix strategies B_3 and B_4, in which case $y_1 = y_2 = 0$ and $y_4 = 1 - y_3$. As a result, B's expected payoffs corresponding to A's pure strategies are given as

A's pure strategy	B's expected payoff
1	$4y_3 - 1$
2	$-4y_3 + 6$

The *best of the worst* solution for B is the minimum point on the *upper* envelope of the given two lines (you will find it instructive to graph the two lines and identify the upper envelope). This process is equivalent to solving the equation

$$4y_3 - 1 = -4y_3 + 6$$

The solution gives $y_3 = \frac{7}{8}$, which yields the value of the game as $v = 4 \times \left(\frac{7}{8}\right) - 1 = \frac{5}{2}$.

The solution of the game calls for player A to mix A_1 and A_2 with equal probabilities and for player B to mix B_3 and B_4 with probabilities $\frac{7}{8}$ and $\frac{1}{8}$. (Actually, the game has alternative solutions for B, because the maximin point in Figure 13.9 is determined by more than two lines. Any non-negative combination of these alternative solutions is also a legitimate solution.)

Remarks. Games in which player A has m strategies and player B has only two can be treated similarly. The main difference is that we will be plotting B's expected payoff corresponding to A's pure strategies. As a result, we will be seeking the minimax, rather than the maximin, point of the *upper envelope* of the plotted lines. However, to solve the problem with TORA, it is necessary to express the payoff in terms of the player that has two strategies by multiplying the payoff matrix by -1, if necessary.

PROBLEM SET 13.4B[4]

1. Solve the coin-tossing game of Example 13.4-2 graphically.

*2. Robin, who travels frequently between two cities, has two route options: Route A is a fast four-lane highway, and route B is a long winding road. The highway patrol has a limited police force. If the full force is allocated to either route, Robin, with her passionate desire for driving "superfast," is certain to receive a $100 speeding ticket. If the force is split 50-50 between the two routes, there is a 50% chance she will get a $100 ticket on route A and only a 30% chance that she will get the same fine on route B. Develop a strategy for both Robin and the police.

[4]The TORA Zero-sum Games module can be used to verify your answer.

3. Solve the following games graphically. The payoff is for Player A.

(a)

	B_1	B_2	B_3
A_1	1	−3	7
A_2	2	4	−6

(b)

	B_1	B_2
A_1	5	8
A_2	6	5
A_3	5	7

4. Consider the following two-person, zero-sum game:

	B_1	B_2	B_3
A_1	5	50	50
A_2	1	1	.1
A_3	10	1	10

(a) Verify that the strategies $\left(\frac{1}{6}, 0, \frac{5}{6}\right)$ for A and $\left(\frac{49}{54}, \frac{5}{54}, 0\right)$ for B are optimal, and determine the value of the game.

(b) Show that the optimal value of the game equals

$$\sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}x_i y_j$$

Linear Programming Solution of Games. Game theory bears a strong relationship to linear programming, in the sense that a two-person zero-sum game can be expressed as a linear program, and vice versa. In fact, G. Dantzig (1963, p. 24) states that J. von Neumann, father of game theory, when first introduced to the simplex method in 1947, immediately recognized this relationship and further pinpointed and stressed the concept of *duality* in linear programming. This section illustrates the solution of games by linear programming.

Player A's optimal probabilities, $x_1, x_2, \ldots$, and x_m, can be determined by solving the following maximin problem:

$$\max_{x_i}\left\{\min\left(\sum_{i=1}^{m} a_{i1}x_i, \sum_{i=1}^{m} a_{i2}x_i, \ldots, \sum_{i=1}^{m} a_{in}x_i\right)\right\}$$

$$x_1 + x_2 + \cdots + x_m = 1$$

$$x_i \geq 0, i = 1, 2, \ldots, m$$

Now, let

$$v = \min\left\{\sum_{i=1}^{m} a_{i1}x_i, \sum_{i=1}^{m} a_{i2}x_i, \ldots, \sum_{i=1}^{m} a_{in}x_i\right\}$$

The equation implies that

$$\sum_{i=1}^{m} a_{ij}x_i \geq v, j = 1, 2, \ldots, n$$

Player A's problem thus can be written as

$$\text{Maximize } z = v$$

subject to

$$v - \sum_{i=1}^{m} a_{ij}x_i \le 0, j = 1, 2, \ldots, n$$

$$x_1 + x_2 + \cdots + x_m = 1$$

$$x_i \ge 0, i = 1, 2, \ldots, m$$

$$v \text{ unrestricted}$$

Note that the value of the game, v, is unrestricted in sign.

Player B's optimal strategies, $y_1, y_2, \ldots,$ and y_n, are determined by solving the problem

$$\min_{y_j}\left\{\max\left(\sum_{j=1}^{n} a_{1j}y_j, \sum_{j=1}^{n} a_{2j}y_j, \ldots, \sum_{j=1}^{n} a_{mj}y_j\right)\right\}$$

$$y_1 + y_2 + \cdots + y_n = 1$$

$$y_j \ge 0, j = 1, 2, \ldots, n$$

Using a procedure similar to that of player A, B's problem reduces to

$$\text{Minimize } w = v$$

subject to

$$v - \sum_{j=1}^{n} a_{ij}y_j \ge 0, i = 1, 2, \ldots, m$$

$$y_1 + y_2 + \cdots + y_n = 1$$

$$y_j \ge 0, j = 1, 2, \ldots, n$$

$$v \text{ unrestricted}$$

The two problems optimize the same (unrestricted) variable v, the value of the game. The reason is that B's problem is the dual of A's problem (verify this claim using the definition of duality in Chapter 4). This means that the optimal solution of one problem automatically yields the optimal solution of the other.

Example 13.4-4

Solve the following game by linear programming.

	B_1	B_2	B_3	Row min
A_1	3	−1	−3	−3
A_2	−2	4	−1	**−2**
A_3	−5	−6	2	−6
Column max	3	4	**2**	

The value of the game, v, lies between -2 and 2.

Player A's Linear Program

$$\text{Maximize } z = v$$

subject to

$$v - 3x_1 + 2x_2 + 5x_3 \leq 0$$
$$v + x_1 - 4x_2 + 6x_3 \leq 0$$
$$v + 3x_1 + x_2 - 2x_2 \leq 0$$
$$x_1 + x_2 + x_3 = 1$$
$$x_1, x_2, x_3 \geq 0$$
$$v \text{ unrestricted}$$

The optimum solution[5] is $x_1 = .39$, $x_2 = .31$, $x_3 = .29$, and $v = -0.91$.

Player B's Linear Program

$$\text{Minimize } z = v$$

subject to

$$v - 3y_1 + y_2 + 3y_3 \geq 0$$
$$v + 2y_1 - 4y_2 + y_3 \geq 0$$
$$v + 5y_1 + 6y_2 - 2y_3 \geq 0$$
$$y_1 + y_2 + y_3 = 1$$
$$v \text{ unrestricted}$$

The solution yields $y_1 = .32$, $y_2 = .08$, $y_3 = .60$, and $v = -0.91$.

PROBLEM SET 13.4C

1. On a picnic outing, 2 two-person teams are playing hide-and-seek. There are four hiding locations (A, B, C, and D), and the two members of the hiding team can hide separately in any two of the four locations. The other team will then have the chance to search any two locations. The searching team gets a bonus point if they find both members of the hiding team. If they miss both, they lose a point. Otherwise, the outcome is a draw.
   *(a) Set up the problem as a two-person zero-sum game.
   (b) Determine the optimal strategy and the value of the game.
2. UA and DU are setting up their strategies for the 1994 national championship college basketball game. Assessing the strengths of their respective "benches," each coach comes up with four strategies for rotating his players during the game. The ability of each team to score 2-pointers, 3-pointers, and free throws is a key factor in determining the final

[5]TORA **Zero-sum Games ⇒ Solve ⇒ LP-based** can be used to solve any two-person zero-sum game.

I M.Sc - OR . Unit: III : Part (2) .

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.

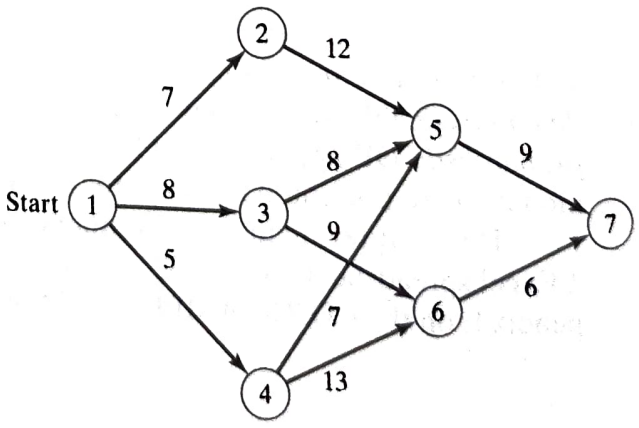
Example 10.1-1 (Shortest-Route Problem)

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.

We can solve this problem by exhaustively enumerating all the routes between nodes 1 and 7 (there are five such routes). However, in a large network, exhaustive enumeration may be intractable computationally.

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



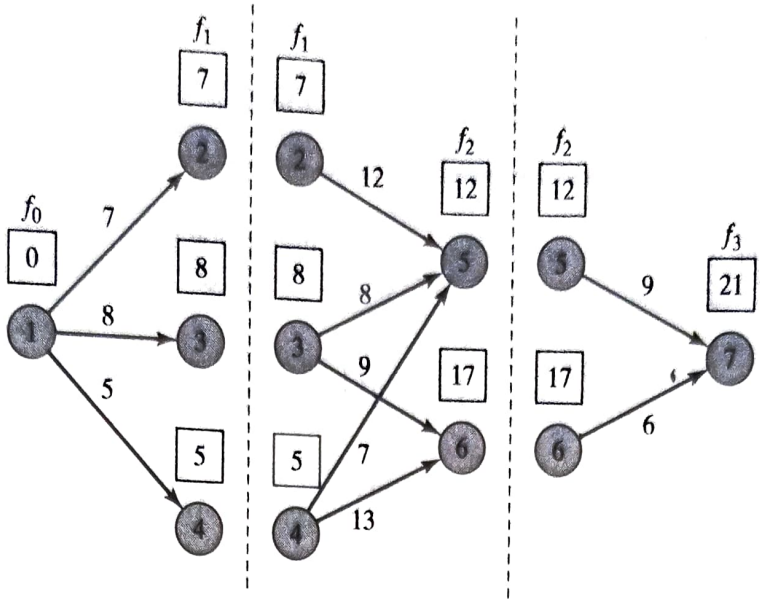


FIGURE 10.2

Decomposition of the shortest-route problem into stages

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 (*from node* 1)
Shortest distance from node 1 to node 3 = 8 miles (*from node* 1)
Shortest distance from node 1 to node 4 = 5 miles (*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

$$\begin{pmatrix}\text{Shortest distance}\\ \text{to node 5}\end{pmatrix} = \min_{i=2,3,4}\left\{\begin{pmatrix}\text{Shortest distance}\\ \text{to node } i\end{pmatrix} + \begin{pmatrix}\text{Distance from}\\ \text{node } i \text{ to node 5}\end{pmatrix}\right\}$$

$$= \min\begin{Bmatrix}7 + 12 = 19\\ 8 + 8 = 16\\ 5 + 7 = 12\end{Bmatrix} = 12 \ (\textit{from node } 4)$$

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

$$\begin{pmatrix}\text{Shortest distance}\\ \text{to node 6}\end{pmatrix} = \min_{i=3,4}\left\{\begin{pmatrix}\text{Shortest distance}\\ \text{to node } i\end{pmatrix} + \begin{pmatrix}\text{Distance from}\\ \text{node } i \text{ to node 6}\end{pmatrix}\right\}$$

$$= \min\begin{Bmatrix}8 + 9 = 17\\ 5 + 13 = 18\end{Bmatrix} = 17 \ (\textit{from node } 3)$$

Stage 2 Summary.

$\therefore$ 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

$$\begin{pmatrix}\text{Shortest distance}\\ \text{to node 7}\end{pmatrix} = \min_{i=5,6}\left\{\begin{pmatrix}\text{Shortest distance}\\ \text{to node } i\end{pmatrix} + \begin{pmatrix}\text{Distance from}\\ \text{node } i \text{ to node 7}\end{pmatrix}\right\}$$

$$= \min\begin{Bmatrix}12 + 9 = 21\\ 17 + 6 = 23\end{Bmatrix} = 21 \ (\textit{from node } 5)$$

Stage 3 Summary.

$\therefore$ 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 $1 \rightarrow 4 \rightarrow 5 \rightarrow 7$.

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:

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

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.

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?

10.2 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

$$f_i(x_i) = \min_{\substack{\text{all feasible} \\ \text{routes } (x_i, x_{i+1})}} \{d(x_i, x_{i+1}) + f_{i+1}(x_{i+1})\}, i = 1, 2, 3$$

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

	$d(x_3, x_4)$	Optimum solution	
x_3	$x_4 = 7$	$f_3(x_3)$	x_4^*
5	9	9	7
6	6	6	7

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:

	$d(x_2, x_3) + f_3(x_3)$		Optimum solution	
x_2	$x_3 = 5$	$x_3 = 6$	$f_2(x_2)$	x_3^*
2	12 + 9 = 21	—	21	5
3	8 + 9 = 17	9 + 6 = 15	15	6
4	7 + 9 = 16	13 + 6 = 19	16	5

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.

	$d(x_1, x_2) + f_2(x_2)$			Optimum solution	
x_1	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$f_1(x_1)$	x_2^*
1	7 + 21 = 28	8 + 15 = 23	5 + 16 = 21	21	4

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.

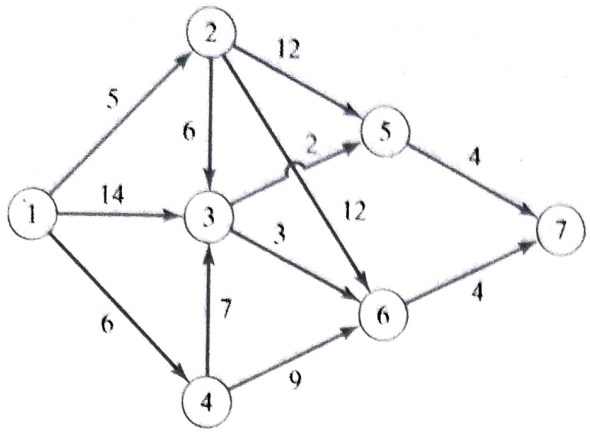


FIGURE 10.3
Network for Problem 3, Set 10.2a

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:

$$\text{Maximize } z = r_1m_1 + r_2m_2 + \cdots + r_nm_n$$

subject to

$$w_1m_1 + w_2m_2 + \cdots + w_nm_n \le W$$
$$m_1, m_2, \ldots, m_n \ge 0 \text{ and integer}$$

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_im_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) $i, i + 1, \ldots$, and 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:

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

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, $x_i - x_{i+1} = w_im_i$ represents the weight used at stage i. Thus, $x_{i+1} = x_i - w_im_i$, and the proper recursive equation is given as

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

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?

Item i	w_i	r_i
1	2	31
2	3	47
3	1	14

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, 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 $\frac{4}{1} = 4$, which means that the possible values of m_3 are 0, 1, 2, 3, and 4. An alternative m_3 is feasible only if $w_3 m_3 \le x_3$. 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.

	$14m_3$					Optimum solution	
x_3	$m_3 = 0$	$m_3 = 1$	$m_3 = 2$	$m_3 = 3$	$m_3 = 4$	$f_3(x_3)$	m_3^*
0	0	—	—	—	—	0	0
1	0	14	—	—	—	14	1
2	0	14	28	—	—	28	2
3	0	14	28	42	—	42	3
4	0	14	28	42	56	56	4

Stage 2. $\max\{m_2\} = \left[\frac{4}{3}\right] = 1$, or $m_3 = 0, 1$

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

	$47m_2 + f_3(x_2 - 3m_2)$		Optimum solution	
x_2	$m_2 = 0$	$m_2 = 1$	$f_2(x_2)$	m_2^*
0	0 + 0 = 0	—	0	0
1	0 + 14 = 14	—	14	0
2	0 + 28 = 28	—	28	0
3	0 + 42 = 42	47 + 0 = 47	47	1
4	0 + 56 = 56	47 + 14 = 61	61	1

Stage 1. $\max\{m_1\} = \left[\frac{4}{2}\right] = 2$ or $m_1 = 0, 1, 2$

$$f_1(x_1) = \max_{m_1=0,1,2}\{31m_1 + f_2(x_1 - 2m_1)\}, \max\{m_1\} = \left[\frac{4}{2}\right] = 2$$

	$31m_1 + f_2(x_1 - 2m_1)$			Optimum solution	
x_1	$m_1 = 0$	$m_1 = 1$	$m_1 = 2$	$f_1(x_1)$	m_1^*
0	0 + 0 = 0	—	—	0	0
1	0 + 14 = 14	—	—	14	0
2	0 + 28 = 28	31 + 0 = 31	—	31	1
3	0 + 47 = 47	31 + 14 = 45	—	47	0
4	**0 + 61 = 61**	**31 + 28 = 59**	**62 + 0 = 62**	**62**	**2**

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 a 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

Success
DATE:
PAGE:

18MMA23C : UNIT - IV

1. Game Theory - 13 ; 13.4

2. Dynamic Programming - 10 ; 10.1, 10.2, 10.3

1 - (i) : Optimal Solution of Two Person Zero Sum Game - 13.4.1

1 - (ii) : Solution of Mixed Stategy Games. 13.4.2.

2 - (i) : Recursive Nature of Computations in DP - 10.1

2 - (ii) : Forward and Backward Recursion - 10.2

2 - (iii) : Selected DP Applications - 10.3.