The Art
and Theory
of Dynamic
Programming

Stuart E. Dreyfuse
Averill M. Law

THE ART AND THEORY
OF DYNAMIC PROGRAMMING

THE ART AND THEORY

OF

DYNAMIC PROGRAMMING

Stuart E. Dreyfus

Department of Industrial Engineering
  and Operations Research
University of California
Berkeley, California

Averill M. Law

Department of Industrial Engineering
University of Wisconsin
Madison, Wisconsin

To Richard Bellman, who taught me much of what I know about dynamic programming, and, more important, whose confidence in me led me to have confidence in myself.

STUART E. DREYFUS

To my wife, Steffi Law, for her patience and understanding during the many months it took to complete this book.

AVERILL M. LAW

This page intentionally left blank

CONTENTS

13 Markovian Decision Processes

14 Stochastic Problems with Linear Dynamics and Quadratic Criteria

15 Optimization Problems Involving Learning

PREFACE

It is our conviction, based on considerable experience teaching the subject, that the art of formulating and solving problems using dynamic programming can be learned only through active participation by the student. No amount of passive listening to lectures or of reading text material prepares the student to formulate and solve novel problems. The student must first discover, by experience, that proper formulation is not quite as trivial as it appears when reading a textbook solution. Then, by considerable practice with solving problems *on his own*, he will acquire the feel for the subject that ultimately renders proper formulation easy and natural. For this reason, this book contains a large number of instructional problems, carefully chosen to allow the student to acquire the art that we seek to convey. The student *must do these problems* on his own. Solutions are given in the back of the book because the reader needs feedback on the correctness of his procedures in order to learn, but any student who reads the solution before seriously attempting the problem does so at this own peril. He will almost certainly regret this passivity when faced with an examination or when confronted with real-world problems. We have seen countless students who have clearly understood and can religiously repeat our lectures on the dynamic programming solution of specific problems fail utterly on midterm examinations asking that the same techniques and ideas be used on similar, but different, problems. Surprised, and humbled, by this experience, the same students often then begin to take seriously our admonition to do homework problems and do not just read the solution and think "of course that is how to do them," and many have written perfect final exams.

Why dynamic programming is more art than science we do not know. We suspect that part of the problem is that students do not expect to be called upon to use common sense in advanced mathematical courses and,

furthermore, have very little practice or experience with common sense in this context. But common sense, not mathematical manipulation, is what it takes to be successful in a course in dynamic programming.

While the preceding remarks are primarily addressed to the reader, the following comments on the organization and use of this text are mainly for the teacher. A certain amount of jargon is used in dynamic programming to express concepts that, while simple, come only from experience. Since we plan to tell the student this jargon only after he has experienced the concept, we believe that it is pointless and hopeless to tell the student, in advance and in a meaningful way, what the text will teach, how it is organized, etc.

Our primary goal is to convey, by examples, the art of formulating the solution of problems in terms of dynamic-programming recurrence relations. The reader must learn how to identify the appropriate state and stage variables, and how to define and characterize the optimal value function. Corollary to this objective is reader evaluation of the feasibility and computational magnitude of the solution, based on the recurrence relation. Secondarily, we want to show how dynamic programming can be used analytically to establish the structure of the optimal solution, or conditions necessarily satisfied by the optimal solution, both for their own interest and as means of reducing computation. Finally, we shall present a few special techniques that have proved useful on certain classes of problems.

We have chosen to treat only deterministic problems in the first eight chapters since illustrative hand calculations are easier and certain subtleties of stochastic problems are avoided. In Chapters 9–14 we have discussed stochastic aspects of many of the earlier topics as well as a few subjects that are purely stochastic. In Chapter 15 a brief introduction to decision problems involving learning is given. The teacher might well reunite the stochastic and learning aspects of various problems with their deterministic forebears. For example, Chapter 1 on elementary deterministic path problems could be followed by Chapter 9 and Chapter 15 (Sections 1–3), also on path problems. Then Chapter 2 on deterministic equipment replacement could be followed by Chapter 10 and Chapter 15 (Section 4). Similarly, Chapter 6 could be followed immediately by Chapter 14 and then Chapter 15 (Section 6).

The complete book contains material that we feel is more than sufficient for a one-semester (15 week) introductory graduate-level course. Since the only prerequisite is a first course in calculus (including partial differentiation) and an elementary knowledge of probability theory (the concept of a density function and of the mean and variance of a random variable), this book is also appropriate for an upper division undergraduate

course. When used as text in such a course of one-quarter (10-week)
duration, we recommend the following, fundamental, material: Chapters
1–4, Chapter 6 (Sections 1–4), Chapters 9–11, Chapter 13, and Chapter 15
(Sections 1–5). When used for a one-semester undergraduate course or a
one-quarter graduate course, the teacher should add to the basic material
listed above whichever additional topics seem most interesting or relevant.

This page intentionally left blank

ACKNOWLEDGMENTS

This page intentionally left blank

ELEMENTARY PATH PROBLEMS

1. Introduction

Dynamic programming is an optimization procedure that is particularly applicable to problems requiring a sequence of interrelated decisions. Each decision transforms the current situation into a new situation. A sequence of decisions, which in turn yields a sequence of situations, is sought that maximizes (or minimizes) some measure of value. The value of a sequence of decisions is generally equal to the sum of the values of the individual decisions and situations in the sequence.

Through the study of a wide variety of examples we hope the reader will develop the largely intuitive skill for recognizing problems fitting the above very general description. We begin with a problem seeking the best path from one physical location to another. Then we elaborate the problem to show how a "situation" may encompass more than just information about location and must be defined in a way that is appropriate to each particular problem.

2. A Simple Path Problem

Suppose for the moment that you live in a city whose streets are laid out as shown in Figure 1.1, that all streets are one-way, and that the numbers shown on the map represent the effort (usually time but some-times cost or distance) required to traverse each individual block. You live at A and wish to get to B with minimum total effort. (In Chapter 4, you will learn how to find minimum-effort paths through more realistic cities.)

You could, of course, solve this problem by enumerating all possible paths from A to B; adding up the efforts, block by block, of each; and then choosing the smallest such sum. There are 20 distinct paths from A to
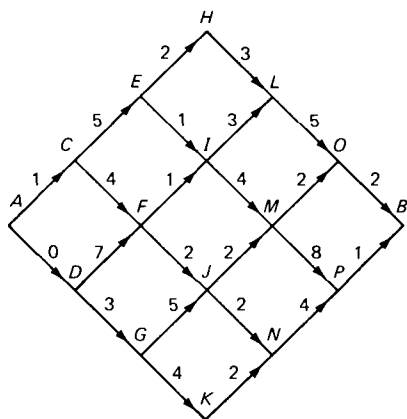
Figure 1.1

B and five additions yield the sum of the six numbers along a particular path, so 100 additions would yield the 20 path sums to be compared. Since one comparison yields the smaller of two numbers, one additional comparison (of that number with a third) yields the smallest of three, etc., 19 comparisons complete this enumerative solution of the problem. As you might suspect, one can solve this problem more efficiently than by brute-force enumeration. This more efficient method is called dynamic programming.

3. The Dynamic-Programming Solution

To develop the dynamic-programming approach, one reasons as follows. I do not know whether to go diagonally upward or diagonally downward from *A*, but if I somehow knew just two additional numbers—namely, the total effort required to get from *C* to *B* by the best (i.e., minimum-effort) path and the total effort required to get from *D* to *B* by the best path—I could make the best choice at *A*. Denoting the minimum effort from *C* to *B* by S_C and the minimum effort from *D* to *B* by S_D, I would add to S_C the effort required in going from *A* to *C*, obtaining the effort required on the best path starting diagonally upward from *A*. I would then add the effort on *AD* to S_D to obtain the effort on the best path starting diagonally downward from *A*, and I would compare these two sums to find the overall minimum effort and the best first decision.

Of course, all this is predicated on knowing the two numbers S_C and S_D which, unfortunately, are not yet known. However, one of the two key ideas of dynamic programming has already made its innocuous appearance. This is the observation that only the efforts along the best

paths from C and from D to B are relevant to the above computation, and the efforts along the nine inferior paths from each of C and D to B need never be computed. This observation is often called the *principle of optimality* and is stated as follows:

The best path from A to B has the property that, whatever the initial decision at A, the remaining path to B, starting from the next point after A, must be the best path from that point to B.

Having defined S_C and S_D as above, we can cite the principle of optimality as the justification for the formula

$$S_A = \min \begin{bmatrix} 1 + S_C \\ 0 + S_D \end{bmatrix},$$

where S_A is the minimum effort to get from A to B and the symbol $\min\begin{bmatrix} x \\ y \end{bmatrix}$ means "the smaller of the quantities x and y." In the future we shall always cite the principle rather than repeat the above verbal reasoning.

Now for the second key idea. While the two numbers S_C and S_D are unknown to us initially, we could compute S_C if we knew the two numbers S_E and S_F (the minimum efforts from E and F to B, respectively) by invoking the principle of optimality to write

$$S_C = \min \begin{bmatrix} 5 + S_E \\ 4 + S_F \end{bmatrix}.$$

Likewise,

$$S_D = \min \begin{bmatrix} 7 + S_F \\ 3 + S_G \end{bmatrix}.$$

S_E, S_F, and S_G are at first not known, but they could be computed if S_H, S_I, S_J, and S_K were available. These numbers, in turn, depend on S_L, S_M, and S_N, which themselves depend on S_O and S_P. Hence we could use formulas of the above type to compute all the S's if we knew S_O and S_P, the minimum efforts from O and P, respectively, to B. But these numbers are trivially known to be 2 and 1, respectively, since O and P are so close to B that only one path exists from each point. Working our way backward from O and P to A, we now carry out the desired computations:

$$S_L = 5 + S_O = 7, \qquad S_M = \min \begin{bmatrix} 2 + S_O \\ 8 + S_P \end{bmatrix} = 4, \qquad S_N = 4 + S_P = 5;$$

$$S_H = 3 + S_L = 10, \qquad S_I = \min \begin{bmatrix} 3 + S_L \\ 4 + S_M \end{bmatrix} = 8, \qquad S_J = \min \begin{bmatrix} 2 + S_M \\ 2 + S_N \end{bmatrix} = 6,$$

$$S_K = 2 + S_N = 7;$$

$$S_E = \min \begin{bmatrix} 2 + S_H \\ 1 + S_I \end{bmatrix} = 9, \qquad S_F = \min \begin{bmatrix} 1 + S_I \\ 2 + S_J \end{bmatrix} = 8,$$

$$S_G = \min \begin{bmatrix} 5 + S_J \\ 4 + S_K \end{bmatrix} = 11;$$

$$S_C = \min \begin{bmatrix} 5 + S_E \\ 4 + S_F \end{bmatrix} = 12, \qquad S_D = \min \begin{bmatrix} 7 + S_F \\ 3 + S_G \end{bmatrix} = 14;$$

$$S_A = \min \begin{bmatrix} 1 + S_C \\ 0 + S_D \end{bmatrix} = 13.$$

Our second key idea has been to compute lengths of the needed minimum-effort paths by considering starting points further and further away from B, finally working our way back to A. Then the numbers required by idea one, the principle of optimality, are known when they are needed.

In order to establish that the best path has total effort 13 (i.e., that $S_A = 13$), we performed one addition at each of the six points H, L, O, K, N, and P where only one decision was possible and we performed two additions and a comparison at each of the remaining nine points where two initial decisions were possible. This sums to 24 additions and nine comparisons, compared with 100 additions and 19 comparisons for the brute-force enumeration described earlier.

Of course we are at least as interested in actually finding the best path as we are in knowing its total effort. The path would be easy to obtain had we noted which of the two possible first decisions yielded the minimum in our previous calculations at each point on the figure. If we let x represent any particular starting point, and denote by P_x the node after node x on the optimal path from x to B, then the P table could have been computed as we computed the S table above. For example, $P_M = O$ since $2 + S_O$ was smaller than $8 + S_P$, $P_I = M$ since $4 + S_M$ was smaller than $3 + S_L$, etc. The P table, which can be deduced with no further computations as the S table is developed, is given in Table 1.1. To use this table to find the best path from A to B we note that $P_A = C$, so we move from A to C. Now, since $P_C = F$, we continue on to F; $P_F = J$ means we move next to J;

Table 1.1 *The optimal next point for each initial point*

$P_O = B$,	$P_P = B$;		
$P_L = O$,	$P_M = O$,	$P_N = P$;	
$P_H = L$,	$P_I = M$,	$P_J = M$,	$P_K = N$;
$P_E = I$,	$P_F = J$,	$P_G = J$ or K;	
$P_C = F$,	$P_D = G$;		
$P_A = C$.			

$P_J = M$ sends us on to M where $P_M = O$ tells us O is next and B is last. The best path is therefore A–C–F–J–M–O–B. As a check on the accuracy of our calculations we add the six efforts along this path obtaining $1 + 4 + 2 + 2 + 2 + 2 = 13$ which equals S_A, as it must if we have made no numerical errors.

It may surprise the reader to hear that there are no further key ideas in dynamic programming. Naturally, there are special tricks for special problems, and various uses (both analytical and computational) of the foregoing two ideas, but the remainder of the book and of the subject is concerned only with how and when to use these ideas and not with new principles or profound insights. What is common to all dynamic-programming procedures is exactly what we have applied to our example: first, the recognition that a given "whole problem" can be solved if the values of the best solutions of certain subproblems can be determined (the principle of optimality); and secondly, the realization that if one starts at or near the end of the "whole problem," the subproblems are so simple as to have trivial solutions.

4. Terminology

To clarify our explanations of various elaborations and extensions of the above ideas, let us define some terms and develop some notations. We shall call the rule that assigns values to various subproblems the *optimal value function*. The function S is the optimal value (here minimum-effort) function in our example. The subscript of S—e.g., the A in the symbol S_A—is the *argument* of the function S, and each argument refers to a particular subproblem. By our definition of S, the subscript A indicates that the best path from A to B is desired, while C means that the best path from C to B is sought. The rule that associates the best first decision with each subproblem—the function P in our example—is called the *optimal policy function*. The principle of optimality yields a formula or set of formulas relating various values of S. This formula is called a *recurrence relation*. Finally, the value of the optimal value function S for certain arguments is assumed obvious from the statement of the problem and from the definition of S with no computation required. These obvious values are called the *boundary conditions* on S.

In this jargon, to solve a problem by means of dynamic programming we choose the arguments of the optimal value function and define that function in such a way as to allow the use of the principle of optimality to write a recurrence relation. Starting with the boundary conditions, we then use the recurrence relation to determine concurrently the optimal value

and policy functions. When the optimal value and decision are known for the value of the argument that represents the original whole problem, the solution is completed and the best path can be traced out using the optimal policy function alone.

We now develop a particular notation for the simple path problem at hand which will allow for a more systematic representation of the procedure than did our earlier formulas. We say nothing new. We shall only say it in a different way. Let us place our city map (Figure 1.1) on a coordinate system as shown in Figure 1.2. Now the point A has coordinates $(0, 0)$, B has coordinates $(6, 0)$, I has $(3, 1)$, etc.

We do not show the one-way arrows on the lines, but we assume throughout the remainder of this chapter that admissible paths are always continuous and always move toward the right.

The optimal value function S is now a function of the pair of numbers (x, y) denoting a starting point, rather than a function of a literal argument such as A or C. For those pairs (x, y) denoting a street intersection on our map (henceforth called a *vertex* of our *network*) we define the optimal value function $S(x, y)$ by

$$S(x, y) = \text{the value of the minimum-effort path connecting}$$
$$\text{the vertex } (x, y) \text{ with the terminal vertex } (6, 0). \qquad (1.1)$$

The diagonal straight line connecting one vertex of our network and a neighboring one represents a block of our city and will now be called an *arc* of our network. We let the symbol $a_u(x, y)$ denote the effort associated with the arc connecting the vertex (x, y) with the vertex $(x + 1, y + 1)$; the subscript u signifies the arc goes diagonally *up* from (x, y). We let $a_d(x, y)$ denote the effort of the arc going diagonally *down* from (x, y) to $(x + 1,$

Figure 1.2

$y - 1$), and we say that $a_u(x, y)$ or $a_d(x, y) = \infty$ (a very large number) if there is no such arc in our network (e.g., $a_u(4, 2) = \infty$).

In terms of these symbols, the principle of optimality gives the recurrence relation

$$S(x, y) = \min \left[\begin{array}{c} a_u(x, y) + S(x + 1, y + 1) \\ a_d(x, y) + S(x + 1, y - 1) \end{array} \right] \tag{1.2}$$

and the obvious boundary condition is

$$S(6, 0) = 0, \tag{1.3}$$

since the effort in going *from* (6, 0) *to* (6, 0) is zero for we are already there. Alternatively, we could write the equally obvious boundary conditions $S(5, 1) = 2$, $S(5, -1) = 1$ as we did earlier, but these are implied by (1.2), (1.3), and our convention that $a_u(5, 1) = \infty$ and $a_d(5, -1) = \infty$. Furthermore, (1.3) is simpler to write. Either boundary condition is correct.

In the exercises assigned during this and subsequent chapters, when we use a phrase such as, "Give the dynamic-programming formulation of this problem," we shall mean:

(1) Define the appropriate optimal value function, including both a specific definition of its arguments and the meaning of the value of the function (e.g., (1.1)).

(2) Write the appropriate recurrence relation (e.g., (1.2)).

(3) Note the appropriate boundary conditions (e.g., (1.3)).

Figure 1.3
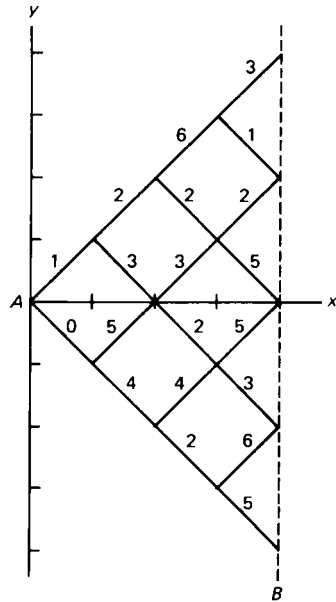
As we emphasized in the Preface, the *art* of dynamic-programming formulation can be mastered only through practice, and it is absolutely essential that the reader attempt almost all of the assigned problems. Furthermore, the student should understand the correct solution given in the back of this book before continuing.

Problem 1.1. On the network shown in Figure 1.3, we seek that path connecting A with any point on line B which minimizes the sum of the four arc numbers encountered along the path. (There are 16 admissible paths.) Give the dynamic-programming formulation of this problem.

Problem 1.2. Solve the above problem using dynamic programming, with the additional specification that there is a rebate associated with each terminal point; ending at the point $(4, 4)$ has a cost of -2 (i.e., 2 is subtracted from the path cost), $(4, 2)$ has cost -1, $(4, 0)$ has cost -3, $(4, -2)$ has cost -4, and $(4, -4)$ has cost -3.

5. Computational Efficiency

Before proceeding, let us pause to examine the efficiency of the dynamic-programming approach to the minimum-effort path problems we have considered. Let us first ask approximately how many additions and comparisons are required to solve a problem on a network of the type first considered, an example of which is shown in Figure 1.4 (without specifying the arc costs and without arrows indicating that, as before, all arcs are directed diagonally to the right). First we note that in the problem represented in Figure 1.1 each admissible path contained six arcs, whereas in the one given in Figure 1.4 each path has 10. We call the former a six-stage problem, the one in Figure 1.4 a 10-stage problem, and we shall now analyze an N-stage problem for N an even integer. There are N vertices (those on the lines CB and DB, excluding B, in Figure 1.4) at
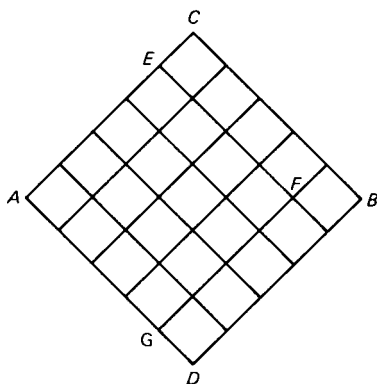
Figure 1.4

which one addition and no comparisons are required in the dynamic-programming solution. There are $(N/2)^2$ remaining vertices (those in the diamond *AEFG*) at which two additions and a comparison are required. Hence a total of $N^2/2 + N$ additions and $N^2/4$ comparisons are needed for the dynamic-programming solution. (Note that for $N = 6$, the first problem in the text, these formulas yield 24 additions and nine comparisons, which checks with the count that we performed earlier.)

When we ask, in future problems, how many additions and how many comparisons are required for the dynamic-programming solution, we expect the reader to do roughly what we did above—imagine that the calculations are really being performed, count the points (or situations) that must be considered, count the additions and comparisons required at each such point (taking account of perhaps varying calculations at differing points), and total the computations.

To get an idea of how much more efficient dynamic programming is than what we called earlier brute-force enumeration, let us consider enumeration for an *N*-stage problem. There are $\binom{N}{N/2}$ admissible paths. (The symbol $\binom{X}{Y}$ should be read, "The number of different ways of choosing a set of *Y* items out of a total of *X* distinguishable items" and $\binom{X}{Y} = X!/[Y!(X - Y)!]$ where $z! = 1 \cdot 2 \cdot 3 \cdots z$.) To derive this formula for the number of paths we note that each path can be represented by a sequence of *N* symbols, half of which are *U*'s and half of which are *D*'s, where a *U* in the *K*th position in the sequence means the *K*th step is diagonally *up* and a *D* means the *K*th step is diagonally *down*. Then $\binom{N}{N/2}$ is the number of different ways of choosing the $N/2$ steps that are *U*, with, of course, the remainder being *D*'s. Note that the formula gives the correct number, 20, for our original six-stage example. Each path requires $N - 1$ additions, and all but the first one evaluated require a comparison in order to find the best path. This totals to $(N - 1)\binom{N}{N/2}$ additions and $\binom{N}{N/2} - 1$ comparisons. For $N = 6$ we have already seen that dynamic programming required roughly one-fourth the computation of brute-force enumeration. However, for $N = 20$ we find that the dynamic-programming solution involves 220 additions and 100 comparisons, while enumeration requires more than three million additions and some 184,000 comparisons. We shall find in general that the larger the problem, the more impressive the computational advantage of dynamic programming.

Problem 1.3. How many additions and how many comparisons are required in the dynamic-programming solution and in brute-force enumeration for an *N*-stage problem involving a network of the type shown in Figure 1.3? Evaluate your formulas for $N = 20$.

Let us note here a further advantage of dynamic programming. Once

a problem has been solved by the computational scheme that we have been using, which works backward from the terminal point or points, one also has solved a variety of other problems. One knows, in our examples, the best paths from each vertex to the end. In our initial example of this chapter, referral to the policy Table 1.1 of Section 3 tells us that the best path from D to B goes first to vertex G, then to J or K, and, if J is chosen, the remaining vertices are M, O, and B.

6. Forward Dynamic Programming

We now explicate a variation on the above dynamic-programming procedure which is equally efficient but which yields solutions to slightly different but related problems. In a sense we reverse all of our original thinking. First we note that we could easily determine the effort of the best path from A to B in Figure 1.1 if we knew the effort of both the best path from A to O and the best path from A to P. Furthermore, we would know these two numbers if we knew the efforts of the best paths to each of L, M, and N from A, etc. This leads us to define a new optimal value function S by

$S(x, y) =$ the value of the minimum-effort path connecting
        the initial vertex $(0, 0)$ with the vertex (x, y).   (1.4)

Note that this is a quite different function from the S defined and computed previously; however, the reader should not be disturbed by our use of the same symbol S as long as each use is clearly defined. There are far more functions in the world than letters, and surely the reader has let $f(x) = x$ for one problem and $f(x) = x^2$ for the next.

The appropriate recurrence relation for our new optimal value function is

$$S(x, y) = \min \begin{bmatrix} a_u(x - 1, y - 1) + S(x - 1, y - 1) \\ a_d(x - 1, y + 1) + S(x - 1, y + 1) \end{bmatrix}. \qquad (1.5)$$

Here we are using a reversed version of the principle of optimality, which can be stated as follows:

The best path from A to any particular vertex B has the property that whatever the vertex before B, call it C, the path must be the best path from A to C.

The boundary condition is

$$S(0, 0) = 0 \qquad (1.6)$$

since the cost of the best path from A to itself is zero.

Problem 1.4. Solve the problem in Figure 1.1 by using (1.4)–(1.6). How many additions and how many comparisons does the solution entail? How does this compare to the numbers for the original dynamic-programming solution in the text?

We shall call the procedure using the new reversed viewpoint the *forward* dynamic-programming procedure since the computation is performed by moving forward from A to B rather than moving *backward* from B to A as in the original (backward) procedure.

The two procedures differ in the auxiliary information they produce. The forward procedure used in Problem 1.4 yields the optimal path *from A* to every vertex, but tells nothing about the optimal paths from most vertices to B. The latter information is furnished by the backward procedure.

Problem 1.5. Do Problem 1.1 by the forward procedure. Compute and compare the number of additions and comparisons for the backward and the forward solutions.

Problem 1.6. Find the minimum-cost path starting from line A and going to line B in the network shown in Figure 1.5. The numbers shown along lines A and B are additional costs associated with various initial and terminal points.
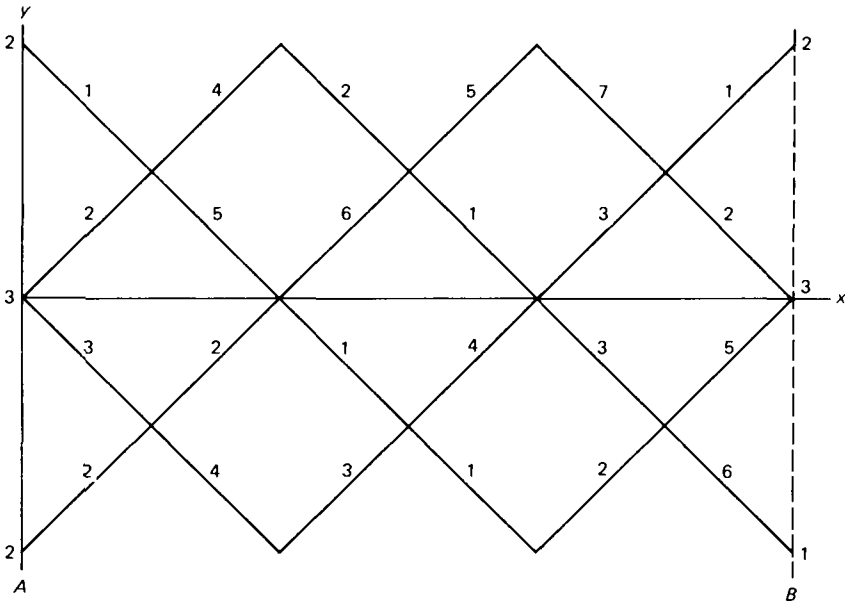
Figure 1.5

7.   A More Complicated Example

We now complicate the situation to show that for some problems the argument of the optimal value function may have to contain more information than just the current vertex. A major part of the art of dynamic programming is the appropriate choice of the subproblems that must be solved in order eventually to solve the given problem.

Let us stipulate that if, after arriving at any vertex on our way from A to B in Figure 1.1, we turn rather than continue in a straight line, an additional cost of 3 is assessed. No penalty is assessed if we continue straight on. The path that solved the original problem before the turn-penalty charge was introduced looked like

and hence had three turns. Therefore, that path costs 22 (13 plus 3 for each of three turns) and may no longer be optimal since there may have been paths originally costing less than 19 and containing one turn.

Suppose, just as we did at the start of this chapter, that we are at the initial vertex A and we already know the cost (including the turn penalties) of the best path from C to B and also from D to B. Can we then compute the optimal decision at A? If we move diagonally upward from A to C, the arc cost is 1, but what is the remaining cost? It is clearly the (assumed known) cost of going from C to B if it is optimal, in going from C to B, to go diagonally upward at C (i.e., continue straight at C rather than turn). But what if the optimal decision at C, in order to minimize the cost including turn penalties of going from C to B, is to go diagonally down at C? Then if we first go from A to C and then follow this path the total cost is the cost of the arc AC plus a cost of 3 for the turn plus the (assumed known) optimal cost from C to B. But, in this case, there is another possibility that we must consider. Maybe, if we were to go diagonally upward to C, we should continue diagonally upward at C and avoid the turn penalty cost of 3. If the minimum-cost path from C to B that starts diagonally upward at C costs only 1 or 2 more than the optimal path from C to B (which we assumed went diagonally down at C), using this path would be preferable to turning at C and going diagonally down, even though the optimal path from C to B goes diagonally down at C. So what we need to know at C is both the minimal cost from C to B, including any turn penalties incurred after C, if we leave C in the diagonally upward direction and also the minimal cost should we choose to leave C in the diagonally downward direction. That is, we need two numbers associated with the vertex C. Let us now apply these insights to solve the problem. We define the optimal value function S, which in this case is a function of

three variables (two describe the vertex and one, which can take on only two possible values, tells us whether we are to leave the vertex by going diagonally up or by going diagonally down), by

$S(x, y, z) =$ the minimum attainable sum of arc numbers plus turn penalties if we start at the vertex (x, y), go to B, and move initially in the direction indicated by z, where z equals 0 denotes diagonally upward and z equals 1 denotes diagonally downward. $\qquad$ (1.7)

Now, supposing we are at (x, y) and move upward $(z = 0)$, we arrive at $(x + 1, y + 1)$ moving in an upward direction. If we continue to move upward, we incur an additional cost $S(x + 1, y + 1, 0)$, but should we turn and proceed downward instead the cost is 3 (the penalty for the turn) plus $S(x + 1, y + 1, 1)$, the remaining cost once we have started downward. The optimal cost from (x, y) starting upward is the minimum of these two alternatives. Hence, by the principle of optimality, we obtain the recurrence relation

$$S(x, y, 0) = \min \begin{bmatrix} a_{\mathrm{u}}(x, y) + S(x + 1, y + 1, 0) \\ a_{\mathrm{u}}(x, y) + 3 + S(x + 1, y + 1, 1) \end{bmatrix}, \qquad (1.8)$$

or, equivalently,

$$S(x, y, 0) = a_{\mathrm{u}}(x, y) + \min \begin{bmatrix} S(x + 1, y + 1, 0) \\ 3 + S(x + 1, y + 1, 1) \end{bmatrix}. \qquad (1.9)$$

Repeating the reasoning above under the assumption that we choose to start from (x, y) in a downward direction, we obtain

$$S(x, y, 1) = a_{\mathrm{d}}(x, y) + \min \begin{bmatrix} 3 + S(x + 1, y - 1, 0) \\ S(x + 1, y - 1, 1) \end{bmatrix}. \qquad (1.10)$$

It is essential to note that we *do not* compare $S(x, y, 0)$ and $S(x, y, 1)$ but, rather, we compute, record, and later use *both* of them. (The clever reader might have observed that if one of the above S values exceeds the other by more than 3, the larger can never be part of an overall minimum-sum path and can be dropped. However, such cleverness, when it exploits the particular data of the problem—the fact that a turn costs 3 in this example —should be avoided since it, at the very least, obscures the general principle and, in this particular example, the computation needed to make the comparison may well exceed the savings accrued from occasionally dropping the larger from consideration.)

Formulas (1.9) and (1.10) together constitute the recurrence relation for this problem. We shall now write (1.9) and (1.10) together as just one formula, but first we want to *discourage* the reader from doing so in the

future and promise not to do so ourselves. The elegance of the result does not, in our opinion, justify the complexity of the formula. By first letting z equal 0 and comparing with (1.9) and then letting z equal 1 and comparing with (1.10), the reader can verify that

$$S(x, y, z) = (1 - z)a_u(x, y) + za_d(x, y)$$

$$+ \min \begin{bmatrix} S(x + 1, y + 1 - 2z, z) \\ 3 + S(x + 1, y + 1 - 2z, 1 - z) \end{bmatrix}. \tag{1.11}$$

The boundary condition for our particular problem is, literally,

$$S(5, 1, 0) = \infty, \qquad S(5, 1, 1) = a_d(5, 1) = 2;$$

$$S(5, -1, 0) = a_u(5, -1) = 1, \qquad S(5, -1, 1) = \infty; \tag{1.12}$$

but if we write

$$S(6, 0, 0) = 0, \qquad S(6, 0, 1) = 0 \tag{1.13}$$

(and use our convention that nonexistent arcs cost ∞), then (1.9) and (1.10) imply the results (1.12). Conditions (1.13) assert that once we are at B, the end, there is no remaining cost no matter what we do there.

8. Solution of the Example

Using (1.7), (1.9) and (1.10), and (1.13), we now actually numerically solve the problem shown in Figure 1.1 with the additional stipulation that each direction change costs 3. $P(x, y, z) = U$ means that *up* is the optimal *second* decision if we start at (x, y) and move first in the direction indicated by z and a similar definition holds for $P(x, y, z) = D$.

Defining S as ∞ at points that are not vertices of our problem network,

$$S(5, 1, 0) = \infty + \min \begin{bmatrix} \infty \\ 3 + \infty \end{bmatrix} = \infty;$$

$$S(5, 1, 1) = 2 + \min \begin{bmatrix} 3 \\ 0 \end{bmatrix} = 2;$$

$$S(5, -1, 0) = 1 + \min \begin{bmatrix} 0 \\ 3 \end{bmatrix} = 1;$$

$$S(5, -1, 1) = \infty + \min \begin{bmatrix} 3 + \infty \\ \infty \end{bmatrix} = \infty.$$

Now, using the above results,

$$S(4, 2, 0) = \infty + \min \begin{bmatrix} \infty \\ 3 + \infty \end{bmatrix} = \infty;$$

$$S(4, 2, 1) = 5 + \min \begin{bmatrix} 3 + S(5, 1, 0) \\ S(5, 1, 1) \end{bmatrix} = 7, \qquad P(4, 2, 1) = D;$$

$$S(4, 0, 0) = 2 + \min\left[\begin{array}{c} S(5, 1, 0) \\ 3 + S(5, 1, 1) \end{array}\right] = 7, \qquad P(4, 0, 0) = D;$$

$$S(4, 0, 1) = 8 + \min\left[\begin{array}{c} 3 + S(5, -1, 0) \\ S(5, -1, 1) \end{array}\right] = 12, \qquad P(4, 0, 1) = U;$$

$$S(4, -2, 0) = 4 + \min\left[\begin{array}{c} S(5, -1, 0) \\ 3 + S(5, -1, 1) \end{array}\right] = 5, \qquad P(4, -2, 0) = U;$$

$$S(4, -2, 1) = \infty.$$

Proceeding to various situations that are three steps from the end (i.e., $x = 3$),

$$S(3, 3, 0) = \infty;$$

$$S(3, 3, 1) = 3 + \min\left[\begin{array}{c} 3 + \infty \\ 7 \end{array}\right] = 10, \qquad P(3, 3, 1) = D;$$

$$S(3, 1, 0) = 3 + \min\left[\begin{array}{c} \infty \\ 3 + 7 \end{array}\right] = 13, \qquad P(3, 1, 0) = D;$$

$$S(3, 1, 1) = 4 + \min\left[\begin{array}{c} 3 + 7 \\ 12 \end{array}\right] = 14, \qquad P(3, 1, 1) = U;$$

$$S(3, -1, 0) = 2 + \min\left[\begin{array}{c} 7 \\ 3 + 12 \end{array}\right] = 9, \qquad P(3, -1, 0) = U;$$

$$S(3, -1, 1) = 2 + \min\left[\begin{array}{c} 3 + 5 \\ \infty \end{array}\right] = 10, \qquad P(3, -1, 1) = U;$$

$$S(3, -3, 0) = 2 + \min\left[\begin{array}{c} 5 \\ \infty \end{array}\right] = 7, \qquad P(3, -3, 0) = U;$$

$$S(3, -3, 1) = \infty.$$

Using these eight numbers to compute all four-step solutions (i.e., $x = 2$),

$$S(2, 2, 0) = 2 + \min\left[\begin{array}{c} \infty \\ 3 + 10 \end{array}\right] = 15, \qquad P(2, 2, 0) = D;$$

$$S(2, 2, 1) = 1 + \min\left[\begin{array}{c} 3 + 13 \\ 14 \end{array}\right] = 15, \qquad P(2, 2, 1) = D;$$

$$S(2, 0, 0) = 1 + \min\left[\begin{array}{c} 13 \\ 3 + 14 \end{array}\right] = 14, \qquad P(2, 0, 0) = U;$$

$$S(2, 0, 1) = 2 + \min\left[\begin{array}{c} 3 + 9 \\ 10 \end{array}\right] = 12, \qquad P(2, 0, 1) = D;$$

$$S(2, -2, 0) = 5 + \min\left[\begin{array}{c} 9 \\ 3 + 10 \end{array}\right] = 14, \qquad P(2, -2, 0) = U;$$

$$S(2, -2, 1) = 4 + \min\left[\begin{array}{c} 3 + 7 \\ \infty \end{array}\right] = 14, \qquad P(2, -2, 1) = U.$$

Next,

$$S(1, 1, 0) = 5 + \min\left[\begin{array}{c} 15 \\ 3 + 15 \end{array}\right] = 20, \qquad P(1, 1, 0) = U;$$

$$S(1, 1, 1) = 4 + \min\left[\begin{array}{c} 3 + 14 \\ 12 \end{array}\right] = 16, \qquad P(1, 1, 1) = D;$$

$$S(1, -1, 0) = 7 + \min\left[\begin{array}{c} 14 \\ 3 + 12 \end{array}\right] = 21, \qquad P(1, -1, 0) = U;$$

$$S(1, -1, 1) = 3 + \min\left[\begin{array}{c} 3 + 14 \\ 14 \end{array}\right] = 17, \qquad P(1, -1, 1) = D.$$

Finally,

$$S(0, 0, 0) = 1 + \min\left[\begin{array}{c} 20 \\ 3 + 16 \end{array}\right] = 20, \qquad P(0, 0, 0) = D;$$

$$S(0, 0, 1) = 0 + \min\left[\begin{array}{c} 3 + 21 \\ 17 \end{array}\right] = 17, \qquad P(0, 0, 1) = D.$$

From the last two numbers we conclude that the cost of the best path starting from A in an upward direction is 20 and in a downward direction is 17. Hence we choose to start in the downward direction. Since $P(0, 0, 1) = D$, we continue in the downward direction at the vertex $(1, -1)$. Since $P(1, -1, 1)$ is D, we continue downward when we reach $(2, -2)$. $P(2, -2, 1)$ being U, we turn and move diagonally upward at $(3, -3)$. $P(3, -3, 0)$ equaling U means we go upward at $(4, -2)$ and $P(4, -2, 0)$ being U means we go upward at $(5, -1)$. The optimal path is therefore $(0, 0)$, $(1, -1)$, $(2, -2)$, $(3, -3)$, $(4, -2)$, $(5, -1)$, $(6, 0)$; and its cost is 14 for its arc costs plus 3 for its one turn, 17 in all, which conforms with our computed value of $S(0, 0, 1)$.

It is crucial for the reader to understand that, to work the problem correctly, the "current situation" must include current direction information as well as current vertex information. Therefore, the optimal value function depends on an additional argument, which in turn somewhat increases the required computations.

Problem 1.7. The preceding problem can equally well be solved by defining the "current situation" to be a vertex and the direction of the arc by which we arrived at that vertex. Give the dynamic-programming formulation using this approach, but you need not solve the problem numerically.

Problem 1.8. For an N-step problem of the above type on a diamond-shaped network, how many additions and how many comparisons are needed for solution? For simplicity, assume that boundary vertices are the same as all others.

Problem 1.9. Give a forward dynamic-programming procedure for solving the turn-penalty problem solved above in the text. Do not solve numerically.

9. The Consultant Question

Once the key ideas of dynamic programming (the use of an optimal value function, its characterization by a recurrence relation, and its recursive solution yielding successively the solutions to problems of longer and longer duration) are understood, the art of dynamic-programming formulation involves the proper choice of the arguments for the optimal value function. If too few arguments are chosen (such as neglecting the direction information in the turn-penalty problem), no correct recurrence relation can be written. If too many are chosen (such as specifying the direction of the next two arcs, rather than one, in the turn-penalty problem) a correct result can be obtained, but at the expense of an unnecessary amount of computation. A good way to determine the right amount of information to incorporate in the arguments of the optimal value function for a backward procedure is to think as follows. Suppose that someone were making a series of (perhaps nonoptimal) decisions for the problem at hand and then decided that he was not doing too well and needed the help of an expert dynamic-programming consultant, namely you, the reader. After he has described the problem to you, but before he tells you anything about what he has done so far, he says, "Now you take over and do things optimally from now on." The minimal information that you would have to acquire from him about the current situation from which you are to start constitutes the arguments of the optimal value function. Thinking of the situation in this way and asking what information you would need to know in order to take over the decision making will subsequently be called "asking the consultant question."

Problem 1.10. Reconsider the original problem of this chapter with no turn penalty (see Figure 1.1) and define the optimal value function by (1.7). Give a correct dynamic-programming formulation based on this unnecessarily elaborate optimal value function.

10. Stage and State

Thus far we have spoken of the arguments of the optimal value function as describing the current situation. Generally one variable describes how many decisions have thus far been made, and it is one larger on the right-hand side of the recurrence relation than on the left regardless of the particular decision. (In (1.2), we are speaking of the variable x; likewise in (1.9) and (1.10). For some definitions of the arguments of the optimal value function, this variable will *decrease* by one after each decision.) This particular monotonic (i.e., either always increasing or always decreasing) variable is called the *stage* variable. All the remaining

variables needed to describe the current situation, given the stage variable, are called *state* variables. The values of the stage and state variables constitute a description of the situation adequate to allow a dynamic-programming solution.

Below is a series of problems of increasing difficulty that will allow the reader to check his understanding of the simple but elusive ideas covered above. The reader who proceeds without verifying that he can correctly solve the problems, *before* reading the solutions at the back of the book, does so at his own peril. It is our experience that there is a wide gap between understanding someone else's dynamic-programming solution and correctly devising one's own, and that the latter skill is acquired only through practice in independent problem solving. Correct problem formulation requires creative thinking rather than merely passive understanding. While we can lead the student to the understanding, we cannot make him think.

Unless otherwise stated, the following 12 problems seek the minimum-cost continuous path connecting A and B, always moving toward the right, on a general N-stage network (N even) of the type shown in Figure 1.6. Each arc has an arc cost associated with it, with $a_u(x, y)$ and $a_d(x, y)$ representing the arc costs as in the text. The total cost of a path is defined in the statement of the problem. For each problem, define the optimal value function, write the recurrence relation and give boundary conditions.
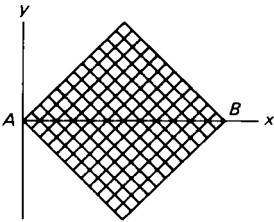
Figure 1.6

Problem 1.11. The cost of a path is the sum of the arc costs plus the total direction-change cost where the kth change of direction costs k, $k = 1, 2, 3, \ldots$ (i.e., a path with three direction changes has a total direction-change cost of $1 + 2 + 3 = 6$).

Problem 1.12. The cost is the sum of the arc costs plus a charge of 3 for each odd-numbered change of direction (i.e., the first, third, fifth, etc. direction change costs 3; other direction changes are free).

Problem 1.13. Only paths with m or fewer direction changes (for given m) are allowable. The cost is the sum of the arc costs. Direction changes cost 0.

Problem 1.14. The cost of each N-stage path from A to B is the sum of the $N - 1$ smallest arc costs along the path (i.e., you do not have to pay the largest arc cost on the path).

Problem 1.15. The cost of a path is the largest arc cost on the path (e.g., the cost of a path with arc costs 1, 7, 6, 2, 5, 3 is 7).

Problem 1.16. The cost of a path is the second largest arc cost on the path.

Problem 1.17. You start at A with Z coupons. If you choose to go diagonally up from (x, y) and simultaneously spend z of your coupons, the original cost $a_u(x, y)$ is reduced by an amount $g_u(x, y, z)$, where g_u increases with increasing z. A similar reduction of $g_d(x, y, z)$ occurs if you go diagonally down. The total cost is the sum of the reduced arc costs.

Problem 1.18. The path may start at any node, not just A. The cost is the sum of the arc costs plus a cost of $p(x, y)$ if the initial node of the path is (x, y).

Problem 1.19. The cost is the usual sum of the arc costs. If you change directions at any vertex, you are not allowed to do so at the next vertex of the path.

Problem 1.20. Only paths with sum of arc costs less than or equal to Z (a given number) are allowed. The cost of a path is the maximum arc cost along the path. All arc costs are nonnegative.

Problem 1.21. All arc costs are positive, and the cost is the *product* of the arc costs along the path.

Problem 1.22. Some arc costs are positive and some are negative. The cost is the product of the arc costs along the path.

11.   The Doubling-Up Procedure

We now use a special kind of simple path problem to develop an idea that, when appropriate, saves considerable computation. The reader will be asked to apply this idea to various problems in subsequent chapters. In what follows we assume that all arcs point diagonally to the right, and we assume, and this is crucial to our method, that while as usual the arc costs depend on their initial and final vertices, they *do not depend* on the stage (i.e., the x coordinate). Such a repeating cost pattern is called stage-invariant and, later, when the stage is often time, it means that costs do not vary with time, only with the nature of the decision. An eight-stage example of such a network with terminal costs (shown in circles) as well as arc costs is shown in Figure 1.7 and subsequently solved.
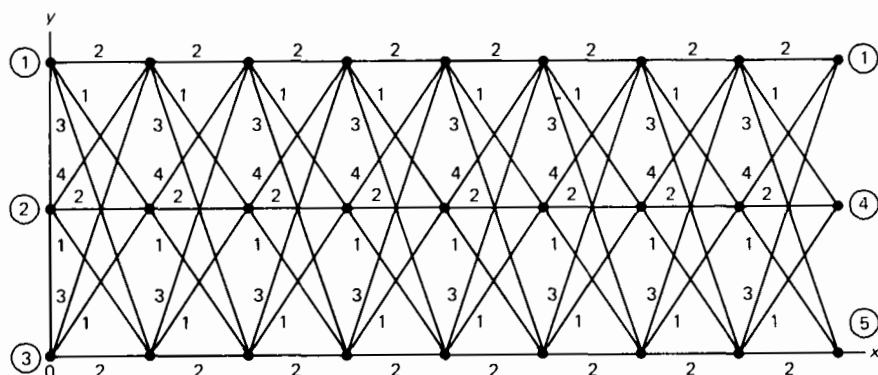
Figure 1.7

Our goal is to devise a procedure for doubling at each iteration the duration of the problem solved. To accomplish this aim we define an optimal value function that depends on three variables as follows:

$S(y_1, y_2, k) =$ the   cost   (ignoring   terminal   costs)   of   the minimum-cost path of length k stages connecting $y = y_1$ and $y = y_2$.   (1.14)

Note that the actual x value at the start or finish is irrelevant, only the number of stages matters. We obtain a recurrence relation for this function by seeking the optimal value of y at the midpoint of a path of duration $2k$ stages connecting y_1 and y_2. Given any particular value of y at the end of k stages of a $2k$-stage problem, we clearly want to proceed *from y_1 to* that value of y in k stages at minimum cost and *from* that value y *to y_2* in the next k stages, also at minimum cost. Then if we minimize this sum over all possible values of y at the end of k stages, we clearly have the cost of the best path of length $2k$ stages. The formula is therefore

$$S(y_1, y_2, 2k) = \min_{y=0,1,2} \left[S(y_1, y, k) + S(y, y_2, k) \right], \qquad (1.15)$$

and the obvious boundary condition is

$$S(y_1, y_2, 1) = c_{y_1 y_2}, \qquad (1.16)$$

where $c_{y_1 y_2}$ is the cost of the single arc directly connecting y_1 to y_2 in one step and, for the example of Figure 1.7, is given by the table

$$c_{00} = 2, \qquad c_{01} = 1, \qquad c_{02} = 3;$$
$$c_{10} = 1, \qquad c_{11} = 2, \qquad c_{12} = 4;$$
$$c_{20} = 3, \qquad c_{21} = 1, \qquad c_{22} = 2.$$

Once $S(y_1, y_2, 8)$ has been computed for all combinations of y_1 and y_2, nine in all, the value of the answer, including terminal costs which we

denote by $t_0(y)$ for $x = 0$ and $t_8(y)$ for $x = 8$, is given by

$$\text{answer} = \min_{\substack{y_1 = 0,1,2 \\ y_2 = 0,1,2}} \left[t_0(y_1) + S(y_1, y_2, 8) + t_8(y_2) \right]. \qquad (1.17)$$

Letting $P(y_1, y_2, 2k)$ denote the optimal decision, i.e., the minimizing *midpoint* on the best path of duration $2k$ stages connecting y_1 and y_2, we now use (1.15)–(1.17) to solve the problem. By (1.16)

$$S(0, 0, 1) = 2, \qquad S(0, 1, 1) = 1, \qquad S(0, 2, 1) = 3,$$
$$S(1, 0, 1) = 1, \qquad S(1, 1, 1) = 2, \qquad S(1, 2, 1) = 4,$$
$$S(2, 0, 1) = 3, \qquad S(2, 1, 1) = 1, \qquad S(2, 2, 1) = 2.$$

Now using (1.15) with $k = 1$ to obtain the optimal two-stage solutions for all pairs of end points,

$$S(0, 0, 2) = \min\left[S(0, 0, 1) + S(0, 0, 1), S(0, 1, 1) + S(1, 0, 1), S(0, 2, 1) \right.$$
$$\left. + S(2, 0, 1) \right]$$
$$= \min[4, 2, 6] = 2, \qquad P(0, 0, 2) = 1;$$
$$S(0, 1, 2) = \min\left[S(0, 0, 1) + S(0, 1, 1), S(0, 1, 1) + S(1, 1, 1), S(0, 2, 1) \right.$$
$$\left. + S(2, 1, 1) \right]$$
$$= \min[3, 3, 4] = 3, \qquad P(0, 1, 2) = 0 \text{ or } 1;$$
$$S(0, 2, 2) = \min[2 + 3, 1 + 4, 3 + 2] = 5, \qquad P(0, 2, 2) = 0, 1, \text{ or } 2;$$
$$S(1, 0, 2) = \min[3, 3, 7] = 3, \qquad P(1, 0, 2) = 0 \text{ or } 1;$$
$$S(1, 1, 2) = \min[2, 4, 5] = 2, \qquad P(1, 1, 2) = 0;$$
$$S(1, 2, 2) = \min[4, 6, 6] = 4, \qquad P(1, 2, 2) = 0;$$
$$S(2, 0, 2) = \min[5, 2, 5] = 2, \qquad P(2, 0, 2) = 1;$$
$$S(2, 1, 2) = \min[4, 3, 3] = 3, \qquad P(2, 1, 2) = 1 \text{ or } 2;$$
$$S(2, 2, 2) = \min[6, 5, 4] = 4, \qquad P(2, 2, 2) = 2.$$

Using (1.15) with $k = 2$ to solve all four-stage problems,

$$S(0, 0, 4) = \min\left[S(0, 0, 2) + S(0, 0, 2), S(0, 1, 2) + S(1, 0, 2), S(0, 2, 2) \right.$$
$$\left. + S(2, 0, 2) \right]$$
$$= \min[2 + 2, 3 + 3, 5 + 2] = 4, \qquad P(0, 0, 4) = 0;$$
$$S(0, 1, 4) = \min[2 + 3, 3 + 2, 5 + 3] = 5, \qquad P(0, 1, 4) = 0 \text{ or } 1;$$
$$S(0, 2, 4) = \min[7, 7, 9] = 7, \qquad P(0, 2, 4) = 0 \text{ or } 1;$$
$$S(1, 0, 4) = \min[5, 5, 6] = 5, \qquad P(1, 0, 4) = 0 \text{ or } 1;$$
$$S(1, 1, 4) = \min[6, 4, 7] = 4, \qquad P(1, 1, 4) = 1;$$
$$S(1, 2, 4) = \min[8, 6, 8] = 6, \qquad P(1, 2, 4) = 1;$$

(equations continue)

$S(2, 0, 4) = \min[4, 6, 6] = 4, \qquad P(2, 0, 4) = 0;$

$S(2, 1, 4) = \min[5, 5, 7] = 5, \qquad P(2, 1, 4) = 0 \text{ or } 1;$

$S(2, 2, 4) = \min[7, 7, 8] = 7, \qquad P(2, 2, 4) = 0 \text{ or } 1.$

Using (1.15) once more to solve all eight-stage problems,

$$S(0, 0, 8) = \min[\, S(0, 0, 4) + S(0, 0, 4),\ S(0, 1, 4) + S(1, 0, 4),\ S(0, 2, 4)$$
$$+ S(2, 0, 4)\,]$$
$$= \min[8, 10, 11] = 8, \qquad P(0, 0, 8) = 0;$$

$S(0, 1, 8) = \min[9, 9, 12] = 9, \qquad P(0, 1, 8) = 0 \text{ or } 1;$

$S(0, 2, 8) = \min[11, 11, 14] = 11, \qquad P(0, 2, 8) = 0 \text{ or } 1;$

$S(1, 0, 8) = \min[9, 9, 10] = 9, \qquad P(1, 0, 8) = 0 \text{ or } 1;$

$S(1, 1, 8) = \min[10, 8, 11] = 8, \qquad P(1, 1, 8) = 1;$

$S(1, 2, 8) = \min[12, 10, 13] = 10, \qquad P(1, 2, 8) = 1;$

$S(2, 0, 8) = \min[8, 10, 11] = 8, \qquad P(2, 0, 8) = 0;$

$S(2, 1, 8) = \min[9, 9, 12] = 9, \qquad P(2, 1, 8) = 0 \text{ or } 1;$

$S(2, 2, 8) = \min[11, 11, 14] = 11, \qquad P(2, 2, 8) = 0 \text{ or } 1.$

Now, using (1.17) to obtain the value of the answer and the optimal choice of the initial and terminal points,

$$\text{answer} = \min[\, t_0(0) + S(0, 0, 8) + t_8(0),\ t_0(0) + S(0, 1, 8) + t_8(1),$$
$$t_0(0) + S(0, 2, 8) + t_8(2),\ t_0(1) + S(1, 0, 8) + t_8(0),$$
$$t_0(1) + S(1, 1, 8) + t_8(1),\ t_0(1) + S(1, 2, 8) + t_8(2),$$
$$t_0(2) + S(2, 0, 8) + t_8(0),\ t_0(2) + S(2, 1, 8) + t_8(1),$$
$$t_0(2) + S(2, 2, 8) + t_8(2)\,]$$
$$= \min[3 + 8 + 5,\ 3 + 9 + 4,\ 3 + 11 + 1,\ 2 + 9 + 5,\ 2 + 8 + 4,$$
$$2 + 10 + 1,\ 1 + 8 + 5,\ 1 + 9 + 4,\ 1 + 11 + 1\,]$$
$$= 13 \text{ with } y_1 = 1,\ y_2 = 2, \text{ and } y_1 = 2,\ y_2 = 2 \text{ both yielding that value.}$$

Using the policy information to reconstruct optimal paths is somewhat tricky. Going from (0, 1) to (8, 2) (i.e., $y_1 = 1$, $y_2 = 2$) we refer to $P(1, 2, 8)$ and find that (4, 1) is the optimal *midpoint*. Now we conclude from $P(1, 1, 4) = 1$ that (2, 1) is the best midpoint of the first four-stage segment, i.e., (2, 1) lies on the path, and from $P(1, 2, 4) = 1$ that 1 is the best midpoint of the second four-stage segment, i.e., that (6, 1) is on the path. Since the first segment connects (0, 1) to (2, 1), we consult $P(1, 1, 2)$, which is 0, to deduce that (1, 0) is on the path. The second segment connects (2, 1) and (4, 1) and $P(1, 1, 2) = 0$ indicates (3, 0) is on the path.

The third segment connects (4, 1) and (6, 1), so again $P(1, 1, 2)$ tells us that (5, 0) is on the path; and, finally, the last segment joins (6, 1) and (8, 2), and $P(1, 2, 2) = 0$ says (7, 0) is on the path. One optimal solution is therefore (0, 1), (1, 0), (2, 1), (3, 0), (4, 1), (5, 0), (6, 1), (7, 0), (8, 2). A similar deductive process tells us that nine paths connect (0, 2) and (8, 2) at a cost of $11 + 2$ for the terminal costs. Just listing the successive y coordinates, they are

2–1–0–1–0–1–0–0–2,   2–1–0–1–0–1–0–1–2,   2–1–0–1–0–1–0–2–2,
2–1–0–1–0–0–1–0–2,   2–1–0–1–0–1–1–0–2,   2–1–0–0–1–0–1–0–2,
2–1–0–1–1–0–1–0–2,   2–1–1–0–1–0–1–0–2,   2–2–1–0–1–0–1–0–2.

Let us now compare the amount of computation required by this *doubling-up* procedure to that required by the usual procedure for this problem, where we assume that the duration is 2^N stages and there are M states at each stage. In our example $N = 3$ and $M = 3$. For doubling-up, each doubling of k requires M additions for each of M^2 pairs (y_1, y_2). We must double-up N times to solve the 2^N-stage problem (neglecting the terminal costs) so $N \cdot M^3$ additions are needed. For the usual procedure, each stage requires M^2 additions (M decisions at each of M points) so roughly $2^N \cdot M^2$ additions are needed. For $N = 3$ and $M = 3$ the usual one-state-variable procedure is slightly better, but for $N = 4$ (i.e., a duration of 16 stages) doubling-up dominates. No matter what M is, for large enough N, doubling-up will dominate. We remind the reader once more that doubling-up can be used only for time-invariant processes, while the usual procedure still works for completely general stage-dependent costs.

To solve using doubling-up, say a 12-stage problem, one can combine $S(y_1, y_2, 8)$ and $S(y_1, y_2, 4)$ by the formula

$$S(y_1, y_2, 12) = \min_y \left[S(y_1, y, 8) + S(y, y_2, 4) \right]. \qquad (1.18)$$

Similarly, a 14-stage problem can then be solved by combining $S(y_1, y_2, 12)$ with $S(y_1, y_2, 2)$, so the procedure can still be used even if the duration is not exactly some power of two. Generally, we have the formula

$$S(y_1, y_2, m + n) = \min_y \left[S(y_1, y, m) + S(y, y_2, n) \right], \qquad (1.19)$$

which raises some interesting questions about the minimum number of iterations to get to some given duration N.

Problem 1.23. Using doubling-up and formulas like (1.18), how many iterations are needed for duration 27? Can you find a procedure using (1.19) that requires fewer iterations?

Chapter 2

EQUIPMENT REPLACEMENT

1. The Simplest Model

We turn now to an application of dynamic programming which is slightly more realistic than life in a one-way city. We consider certain simple equipment-replacement models, dealing with the most elementary situations in the text and somewhat more complex situations in the problems. The reader concerned with navigating a city with two-way streets and with more general path problems is referred to Chapter 4, although we prefer that he continue to develop his intuition and art with the typical dynamic-programming problems in the next two chapters prior to learning the rather specialized procedures that are most efficient for general path problems. More realistic equipment-replacement models than those of this chapter will appear in Chapter 10 where stochastic (risky) situations are treated.

Our basic problem concerns a type of machine (perhaps an automobile) which deteriorates with age, and the decision to replace it. We assume that we must own such a machine during each of the next N time periods (called years in what follows, although the same ideas solve problems with monthly, daily, etc. decisions) and that the cost of operating a machine for one year is a known quantity and depends on the age of the machine. Whenever a machine gets intolerably old, we may replace it, paying a known amount for a new machine and receiving in return a given amount, which depends upon the age of our old machine, as a trade-in value. When the process ends after N years, we receive a salvage value which depends upon the age of our old machine.

Hence, to define our problem, we must be told the duration N of the process, the age y of the machine (called the incumbent machine) with which we start the process, and the following data:

$c(i)$ = cost of operating for one year a machine which is of age i at the start of the year,

p = price of a new machine (of age 0),

$t(i)$ = trade-in value received when a machine which is of age i at the start of a year is traded for a new machine at the start of the year,

$s(i)$ = salvage value received for a machine that has just turned age i at the end of year N.

The problem is to decide when (or if) to replace the incumbent machine, when (or if) to replace its replacement, etc., so as to minimize the total cost during the next N years.

2. Dynamic-Programming Formulation

We now have the opportunity to test what we learned in Chapter 1 on a "real" problem. Recall that first an optimal value function appropriate to the problem must be defined and that the arguments of the function should furnish that information which you, as a consultant, would have to elicit from a decision maker who had been replacing his own machine from time to time and then decides to hire you.

Problem 2.1. Define the appropriate optimal value function for the backward dynamic-programming solution of the above problem and write the recurrence relation and boundary condition that solve the problem.

Do not proceed until you have read the solution of Problem 2.1. The material is essential and is only relegated to the back of the book to encourage the reader to solve the problem before reading the solution.

Now, assuming that you have read and understood the solution of Problem 2.1, consider the problem given by the following data:

$N = 5;$

y (the age of the incumbent machine at the start of year 1) = 2;

$c(0) = 10,\quad c(1) = 13,\quad c(2) = 20,\quad c(3) = 40,$

$c(4) = 70,\quad c(5) = 100,\quad c(6) = 100;$

$p = 50;$

$t(1) = 32,\quad t(2) = 21,\quad t(3) = 11,\quad t(4) = 5,\quad t(5) = 0,\quad t(6) = 0;$

$s(1) = 25,\quad s(2) = 17,\quad s(3) = 8,\quad s(4) = 0,\quad s(5) = 0,\quad s(7) = 0.$

Problem 2.2. Use the solution of Problem 2.1 to solve the above problem numerically. What is the optimal sequence of decisions?

Problem 2.3. For a problem of the above type, approximately how many additions and how many comparisons, as a function of the duration of the process N, are required for the dynamic-programming solution?

We shall assume henceforth that a modern digital computer takes 10^{-5} seconds to perform either an addition or comparison (i.e., it can add 100,000 numbers/second). Actually machines are somewhat faster than this, but several machine operations are required for the logic of locating the appropriate numbers and other bookkeeping procedures.

For the above problem, even if we make monthly decisions over a 20-year horizon (i.e., $N = 12 \cdot 20 = 240$), the computer solution by dynamic programming would require only about $\frac{1}{2} \cdot 240 \cdot 241 \cdot 10^{-5}$ seconds, or approximately $1\frac{1}{2}$ seconds.

While we shall not continue to belabor the advantage of dynamic programming over brute-force enumeration, we cannot resist one more comparison. There are 2^N possible decision sequences for an N-period problem, so for $N = 240$, the solution would take about $2^{240} \cdot 240 \cdot 10^{-5}$ seconds which is more than 10^{69} seconds or about 10^{62} years. This is much larger than the age of our solar system.

Problem 2.4. Do Problem 2.1 using the *forward* dynamic-programming procedure. Approximately how many additions and comparisons does the solution require?

3. Shortest-Path Representation of the Problem

We hope that the reader has become comfortable with viewing the numerical solution of a problem as the sequential computation and tabulation of functions, each one representing the optimal value function for a given number of stages. For any reader who still finds this perplexing but who was at ease solving the path problems of Chapter 1, we now show how the above equipment-replacement model can be viewed as a minimum-cost path problem. Almost all dynamic-programming problems can be thought of as problems seeking the minimum-cost path (generally in more than two dimensions and therefore generally not easily drawn). Letting the x axis denote the year and the y axis represent the age of the machine, we start at $(1, y)$. The "buy" decision takes us to $(2, 1)$ at an arc cost of $p - t(y) + c(0)$ and "keep" leads us to $(2, y + 1)$ with an arc cost of $(c(y))$. The same reasoning applies at each point. For the data of Problem 2.2, we obtain the network given in Figure 2.1, on which all diagonal arcs are directed to the right due to the inexorable increase in time.

From the figure one can easily verify that the paths denoted by *BBKBK* (where a *B* in the *k*th position of a sequence means "buy at the
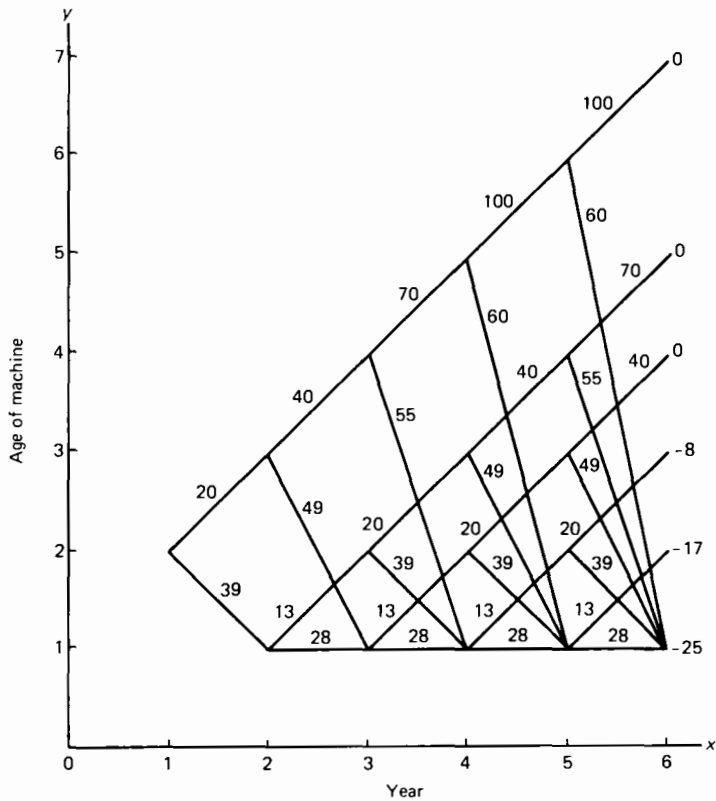
Figure 2.1

start of year k" and, similarly, K denotes "keep") and $BKBBK$ indeed cost 115, as calculated in Problem 2.2.

Problem 2.5. Copy Figure 2.1 and use dynamic programming, doing the computations mentally, to write the value of the optimal value function at each vertex. Denote the optimal decision by an arrow pointing to the appropriate choice of next vertex. Verify that all values and arrows conform with the tabular solution of Problem 2.2.

4. Regeneration Point Approach

The equipment-replacement model that we are studying has a property that, while not common, occurs often enough (for example, see Chapter 12, Section 3) to merit special treatment. Unlike many path

problems, for this type of problem we can be sure in advance that all paths, but one, eventually return at least once to a vertex on the horizontal line $y = 1$ (i.e., sooner or later we buy a new machine, unless we keep the incumbent for the duration of the process). When we return to $y = 1$ the process is said to "regenerate" itself and we can ask the question, "Given an initial situation, when shall we make our first purchase?" To solve the problem recursively, based on this question, we define the optimal value function $S(i)$ as

$$S(i) = \text{the minimum attainable cost for the remaining process given we start year } i \text{ with a one-year-old machine.} \tag{2.1}$$

We use a version of the principle of optimality, still based on common sense, that whenever we choose next to purchase a new machine, the remaining decisions after that purchase must yield the minimum cost for the remaining process, starting with that new machine. This produces the recurrence relation

$$S(i) = \min \begin{bmatrix} \sum_{k=1}^{N-(i-1)} c(k) - s(N - i + 2) \\ \min_{j=i,\ldots,N} \left\{ \sum_{k=1}^{j-i} c(k) + p - t(j - i + 1) + c(0) + S(j + 1) \right\} \end{bmatrix}. \tag{2.2}$$

The top line on the right evaluates the cost of keeping the incumbent until the end of year N. The variable j in the second line is the year of the next purchase and the line adds the cost accrued by the incumbent, the first-year cost of the next machine, and the minimum cost of starting year $j + 1$ with a one-year-old machine. The boundary condition is

$$S(N + 1) = -s(1). \tag{2.3}$$

Using this procedure to solve for the data shown in Figure 2.1, we get
$S(6) = -25;$

$$S(5) = \min \begin{bmatrix} 13 - 17 \\ 28 + S(6) \end{bmatrix} = -4,$$

$P(5) = $ keep until end;

$$S(4) = \min \begin{bmatrix} 13 + 20 - 8 \\ 28 + S(5), \ 13 + 39 + S(6) \end{bmatrix} = 24,$$

$P(4) = $ buy at start of year 4;

$$S(3) = \min \begin{bmatrix} 13 + 20 + 40 \\ 28 + S(4), \ 13 + 39 + S(5), \ 13 + 20 + 49 + S(6) \end{bmatrix} = 48,$$

$P(3) = $ buy at start of year 4;

$$S(2) = \min \begin{bmatrix} 13 + 20 + 40 + 70 \\ 28 + S(3), \ 13 + 39 + S(4), \ 13 + 20 + 49 + S(5), \\ 13 + 20 + 40 + 55 + S(6) \end{bmatrix} = 76,$$

$P(2) =$ buy at start of either year 2 or year 3.

The answer is

$$\min \begin{bmatrix} 20 + 40 + 70 + 100 + 100 \\ 39 + S(2), \ 20 + 49 + S(3), \ 20 + 40 + 55 + S(4), \\ 20 + 40 + 70 + 60 + S(5), \ 20 + 40 + 70 + 100 + 60 + S(6) \end{bmatrix} = 115.$$

The decision is to buy at once.

Using the policy information to determine the optimal sequence of decisions, we naturally obtain the same two optimal policies as before.

This method computes only one number per stage, rather than a table of numbers. Each computation, however, involves an increasing number of alternatives and each alternative an increasing number of additions as we consider processes starting further and further from the end. If done as above the number of additions is of order N^3, but with some cleverness we can reduce the number to approximately the number required by earlier methods. Examining the bottom line on the right of (2.2), we can see that the cost of a particular value of j can be gotten from the cost of the previous (one smaller) value of j by adding one additional c term, by adding back the t term and subtracting the S term of the previous j value, and then by subtracting the t term and adding the S term called for in the formula.

Problem 2.6. For the problem of Section 1, suppose any machine reaching age M (M given) must be replaced and, furthermore, that one can trade-in a used machine on a machine of any age between 0 (new) and $M - 1$. The cost of replacing an i-year-old machine by one of age j is $u(i, j)$, where $u(i, 0) = p - t(i)$ and $u(i, i)$ (the cost of keeping the current machine) equals 0. Give the backward dynamic-programming solution of the problem and determine the approximate number of additions and comparisons.

Problem 2.7. Suppose in Problem 2.6 that all of the data are year-dependent; i.e., that costs in year k differ from those in year j for $j \neq k$. Let a subscript on the data denote the year; e.g., $u_5(4,2) =$ cost of trading a 4-year-old machine for a 2-year-old one at the start of year 5, etc. Give the backward dynamic-programming solution.

5. More Complex Equipment-Replacement Models

We now complicate the problem in a more fundamental way, thereby requiring a new definition of the optimal value function. Afterward, we

shall test the reader's ability along these lines with a series of similar problems of increasing difficulty.

Suppose we now return to the "keep" or "replace" assumption, but assume that one additional decision is available, namely, "overhaul." An overhauled machine is better than one not overhauled, but not as good as a new one. Let us further assume that performance depends on the actual age of equipment and on the number of years since last overhaul, but is independent of when and how often the machine was overhauled prior to its last overhaul. The known data are

$e(k, i, j)$ = cost of exchanging a machine of age i, last overhauled at age j, for a new machine at the start of year k;

$c(k, i, j)$ = net operating cost during year k of a machine of age i, last overhauled at age j;

$o(k, i)$ = cost of overhauling a machine of age i at start of year k;

$s(i, j)$ = salvage value at the end of year N of a machine which has just become age i, last overhauled at age j.

We agree that if j (age at last overhaul) equals 0, the machine has never been overhauled.

If a consultant were hired at the start of year k to make optimal decisions thereafter, he would need to know the age and age at last overhaul of the current machine in order to know how to proceed. Hence, we define

$f(k, i, j)$ = the cost during the remaining process given we start year k with a machine of age i years, last overhauled at age j years, and pursue an optimal replacement policy.   (2.4)

(We have intentionally used f to denote the optimal value function and have written its arguments in a different order so as to impress upon the reader the arbitrary nature of these choices.) The recurrence relation is

$$f(k, i, j) = \min \begin{bmatrix} \text{Replace:} & e(k, i, j) + c(k, 0, 0) + f(k + 1, 1, 0) \\ \text{Keep:} & c(k, i, j) + f(k + 1, i + 1, j) \\ \text{Overhaul:} & o(k, i) + c(k, i, i) + f(k + 1, i + 1, i) \end{bmatrix}$$

(2.5)

and the boundary condition is

$$f(N + 1, i, j) = -s(i, j).$$ (2.6)

For $k = N$, assuming the incumbent machine is new, we must compute f for $i = 1, 2, \ldots, N - 1$ and $j = 0, 1, 2, \ldots, i - 1$. (Note that

starting any year, age at last overhaul is less than age.) This involves $N - 1$ evaluations of three decisions for $i = N - 1$, $N - 2$ for $i = N - 2, \ldots$, and 1 for $i = 1$, a total of $(N - 1)(N)/2$. Then for $k = N - 1$ we have $(N - 2)(N - 1)/2$ such evaluations, for $k = N - 2$ we have $(N - 3)(N - 2)/2$ such evaluations, etc. The total number is precisely $\sum_{i=2}^{N}[(i - 1)i/2] + 1$, or approximately $\sum_{i=1}^{N} i^2/2 \approx N^3/6$. Consequently, the total number of operations is roughly N^3 since each evaluation of the right-hand side of (2.5) requires a total of seven additions and comparisons.

Now suppose we allow exchange for a machine of any age up to a fixed number m and any age at last overhaul less than the age of the acquired machine. There are now roughly $m^2/2$ decisions, rather than three, on the right-hand side of the recurrence relation.

For $k = N$, f must be evaluated for roughly $(m + N)^2/2$ combinations of i and j, $\ldots$; for $k = 2$ there are roughly $m^2/2$ such combinations. Hence, solution requires roughly $(m^2/12)[(m + N)^3 - m^3]$ evaluations of individual decisions. Scheduling by months (rather than years) for, say, 100 months and letting $m = 50$, about 675 million evaluations are required, or roughly two hours of calculation on a high-speed computer.

This last model illustrates how a problem can prove very time-consuming and often intractible for even so efficient a computational scheme as dynamic programming.

The following exercises should allow the reader to test his ability to choose stage and state variables properly, write recurrence relations and boundary conditions, and determine the approximate number of computations required.

Problem 2.8. Consider the model of Section 1 with the additional option "sell your current machine if you own it and lease a new machine for the year." If you already have a leased machine, you can lease again or buy a new machine at cost p. If you sell your machine but do not buy one, you get the salvage value $s(i)$. Let l be the cost of leasing a machine for a year, excluding operating cost. Assuming that you start year 1 with an owned, new machine, give a backward dynamic-programming solution procedure. Approximately how many operations are needed for numerical solution?

Problem 2.9. Solve the problem of Section 1, subject to the additional constraint that after trading your original machine you must trade your current machine at or before the age at trade-in of the previous machine. That is, if x_i ($i > 1$) is the age at trade-in of the ith machine that you trade, then $x_{i+1} \leqslant x_i$.

Problem 2.10. In the model with overhaul (at the beginning of Section 5) suppose that a machine produces a positive net *revenue* $n(k, i, j)$ rather than a *cost* $c(k, i, j)$. Net revenue plus salvage minus exchange and overhaul costs is to be maximized. Give a dynamic-programming formulation.

Problem 2.11. Solve Problem 2.10 assuming that an overhaul requires 2 years, during which time the machine ages but there is no revenue. An overhaul cannot be commenced at the start of year N.

Problem 2.12. Consider the maximum-revenue model of Problem 2.10. Let $C(k)$ denote the capital position at the start of year k. $C(1) = 0$ and $C(k + 1) = C(k)$ plus net revenue during year k minus exchange or overhaul costs during year k. You wish to maximize $C(N + 1)$ plus salvage value subject to the constraint that the capital position during the process must always be nonnegative and exchange or overhaul costs are paid each period *before* obtaining revenue. Give both a backward and a forward dynamic-programming solution.

Problem 2.13. Consider the model of Problem 2.6. Let the duration of the process be 2^N and give a "doubling-up of the number of stages" procedure similar to that of Chapter 1, Section 11. In terms of N and M defined above, compare the number of computations for your procedure to that of the solution of Problem 2.6.

Chapter 3

RESOURCE ALLOCATION

1. The Simplest Model

The basic model of this chapter is of practical interest and also illustrates the application of dynamic programming to a problem that is not really dynamic (i.e., requiring a *sequence* of decisions). It is also the vehicle for developing one of the subject's more important special-purpose computational devices, a novel use of Lagrange multipliers.

The simplest resource allocation problem is as follows. You are given X units of a resource and told that this resource must be distributed among N activities. You are also given N data tables $r_i(x)$ (for $i = 1, \ldots, N$ and $x = 0, 1, \ldots, X$) representing the return realized from an allocation of x units of resource to activity i. (We assume, unless stated otherwise, that all return functions are nondecreasing functions of their arguments.) The problem is to allocate all of the X units of resource to the activities so as to maximize the total return, i.e., to choose N nonnegative integers x_i, $i = 1, \ldots, N$, that maximize

$$\sum_{i=1}^{N} r_i(x_i) \tag{3.1}$$

subject to the constraint

$$\sum_{i=1}^{N} x_i = X. \tag{3.2}$$

2. Dynamic-Programming Formulation

Although, as stated, the choice of the solution set x_i, $i = 1, \ldots, N$, seems to be a single decision made at a particular time, to use dynamic programming we view the problem differently. Having designated the

activities by arbitrary but fixed numbers from 1 to N, we take the X units of resource and first allocate an amount x_1 to activity 1. Then we allocate x_2 units of the remaining $X - x_1$ units of resource to activity 2, then x_3 units of resource to activity 3, etc. Viewed in this sequential fashion, we can ask the "consultant question" (see Chapter 1, Section 9). Assuming that several allocations have already been made, our hypothetical consultant would need to know which activity the decision maker was now ready to consider (or, equivalently, how many activities had already received their allocations) and how much resource was still unallocated. While it would not hurt to know what each of the previously allocated activities received, only the sum already allocated, and hence the amount left, is essential to our optimization of the remaining process. Hence we define the optimal value function by

$f_k(x) =$ the maximum return obtainable from activities k
         through N, given x units of resource remaining to be
         allocated.   (3.3)

As is frequently done in dynamic programming, we have this time chosen to write the stage variable k as a subscript to distinguish it from the other (state) variable. This is a quite arbitrary matter and $S(x, k)$ or a myriad of other notations would be equally acceptable as long as the definition of the optimal value function tells us what each argument means. For that matter the state variable could be defined as the total allocation *thus far* made and/or k could be defined as the number of activities *thus far* considered as long as the recurrence relation was appropriate to the definition.

By the principle of optimality, the recurrence relation appropriate to definition (3.3) is

$$f_k(x) = \max_{x_k = 0, 1, \ldots, x} \left[r_k(x_k) + f_{k+1}(x - x_k) \right],\qquad (3.4)$$

where x_k is the allocation to activity k and $f_k(x)$ must be computed for $x = 0, \ldots, X$. The boundary condition is

$$f_N(x) = r_N(x).\qquad (3.5)$$

The answer is $f_1(X)$.

3. Numerical Solution

We now solve a small hypothetical problem with $X = 8$ and $N = 4$, although we trust the reader can by now translate mathematical formulation into computational solution on his own. The data for this problem are given in Table 3.1. We use the symbol $p_k(x)$ to denote the optimal allocation to activity k if we have x units of resource to distribute to activities $k, k + 1, \ldots, N$. Hence $p_k(x)$ is the value of x_k that maximizes the right-hand side of (3.4). Boundary condition (3.5) gives

Table 3.1.   *Data for the hypothetical problem*

x_1	$r_1(x_1)$	x_2	$r_2(x_2)$	x_3	$r_3(x_3)$	x_4	$r_4(x_4)$
0	0	0	0	0	0	0	0
1	3	1	1	1	2	1	1
2	7	2	2	2	4	2	3
3	10	3	4	3	6	3	6
4	12	4	8	4	8	4	9
5	13	5	13	5	10	5	12
6	14	6	17	6	12	6	14
7	14	7	19	7	14	7	16
8	14	8	20	8	16	8	17

$$f_4(0) = 0, \qquad f_4(1) = 1, \qquad f_4(2) = 3, \qquad f_4(3) = 6, \qquad f_4(4) = 9,$$
$$f_4(5) = 12, \qquad f_4(6) = 14, \qquad f_4(7) = 16, \qquad f_4(8) = 17.$$

Recurrence relation (3.4) yields

$$f_3(0) = 0, \qquad p_3(0) = 0;$$
$$f_3(1) = \max[0 + 1, 2 + 0] = 2, \qquad p_3(1) = 1;$$
$$f_3(2) = \max[0 + 3, 2 + 1, 4 + 0] = 4, \qquad p_3(2) = 2;$$
$$f_3(3) = \max[0 + 6, 2 + 3, 4 + 1, 6 + 0] = 6, \qquad p_3(3) = 0 \text{ or } 3;$$
$$f_3(4) = \max[0 + 9, 2 + 6, 4 + 3, 6 + 1, 8 + 0] = 9, \qquad p_3(4) = 0;$$
$$f_3(5) = \max[0 + 12, 2 + 9, 4 + 6, 6 + 3, 8 + 1, 10 + 0] = 12, \qquad p_3(5) = 0;$$
$$f_3(6) = 14, \qquad p_3(6) = 0 \text{ or } 1;$$
$$f_3(7) = 16, \qquad p_3(7) = 0, 1, \text{ or } 2;$$
$$f_3(8) = 18, \qquad p_3(8) = 1, 2, \text{ or } 3.$$

$$f_2(0) = 0, \qquad p_2(0) = 0;$$
$$f_2(1) = \max[0 + 2, 1 + 0] = 2, \qquad p_2(1) = 0;$$
$$f_2(2) = \max[0 + 4, 1 + 2, 2 + 0] = 4, \qquad p_2(2) = 0;$$
$$f_2(3) = \max[0 + 6, 1 + 4, 2 + 2, 4 + 0] = 6, \qquad p_2(3) = 0;$$
$$f_2(4) = 9, \qquad p_2(4) = 0;$$
$$f_2(5) = 13, \qquad p_2(5) = 5;$$
$$f_2(6) = 17, \qquad p_2(6) = 6;$$
$$f_2(7) = 19, \qquad p_2(7) = 6 \text{ or } 7;$$
$$f_2(8) = 21, \qquad p_2(8) = 6 \text{ or } 7.$$

$$f_1(8) = \max[0 + 21, 3 + 19, 7 + 17, 10 + 13, 12 + 9, 13 + 6, 14 + 4,$$
$$14 + 2, 14 + 0] = 24, \qquad p_1(8) = 2.$$

(We need not compute $f_1(7)$, $f_1(6)$, etc. since we are told that we start with 8 units of resource.)

The optimal return from an 8-unit total allocation is 24. Since $p_1(8) = 2$, we allocate 2 units of resource to activity 1, leaving 6. Since $p_2(6) = 6$, we allocate 6 to activity 2, leaving 0. We can check that $r_1(2) + r_2(6) = 24 = f_1(8)$.

4. Miscellaneous Remarks

We can represent this problem as a longest-path problem as shown in Figure 3.1 for $N = 4$ and $X = 4$ (to draw $X = 8$ would require too many lines). The number on the arc from (k, x) to $(k + 1, x - y)$ (which we show only for stage 1) is $r_k(y)$ and the number associated with the terminal point $(4, x)$ is $r_4(x)$. Now the problem can be solved as in Chapter 1, with maximum replacing minimum.

The computational procedure requires $(N - 2)[(X + 1)(X + 2)/2] + (X + 1)$ additions and an almost equal number of comparisons. This is because consideration of activities k through N requires $X + 1$ additions for $x = X$, X additions for $x = X - 1, \ldots,$ and finally one addition for $x = 0$. This sums to $(X + 1)(X + 2)/2$ additions and must be done for $k = N - 1$, then $N - 2, \ldots,$ and finally $k = 2$. Then $X + 1$ additions are needed for $k = 1$. For $X = 100$ and $N = 12$, we need roughly 50, 000 additions. This would require about one second on a modern high-speed digital computer, allowing for an equal number of comparisons and assuming that either operation takes roughly 10^{-5} seconds.
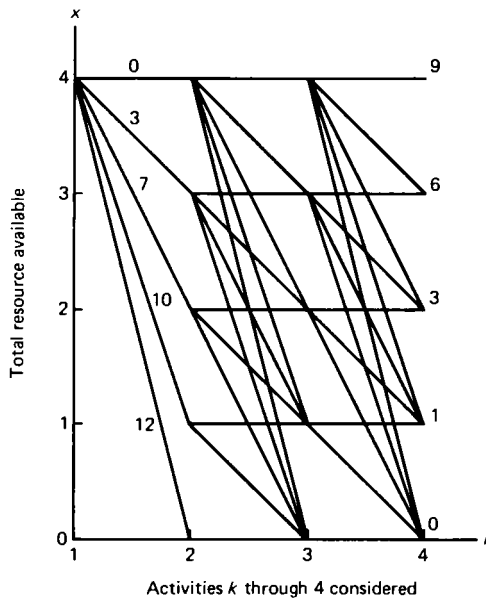
Figure 3.1

There are $\binom{X+N-1}{N-1}$ different allocation policies (this formula is not trivially obvious and its derivation is beyond the scope of this book), each requiring $N-1$ additions to evaluate. Consequently, direct enumeration for $X = 100$ and $N = 12$ would necessitate more than 1,000,000,000,000,000 additions. This would require roughly 300 years on a modern high-speed digital computer.

Problem 3.1. Give the forward dynamic-programming procedure for the above problem and use it to solve the problem numerically. Approximately how many additions does the forward procedure require?

Problem 3.2. Suppose the return functions $r_k(x_k)$ are not necessarily nondecreasing functions of x_k and that (3.2) is replaced by the inequality $\sum_{i=1}^{N} x_i \leqslant X$. How does this affect the dynamic-programming solution?

Problem 3.3. Suppose that there are eight activities and each has the same return data. Give a doubling-up procedure. (See Chapter 1, Section 11.) You need only one stage and one state variable if you use the fact that the optimal return for a k-activity problem depends only on the total amount allocated and not explicitly on the initial and terminal number of units of resource (e.g., going optimally from initial resource 10 to terminal resource 3 yields the same return as proceeding optimally from initial resource 9 to terminal resource 2). Use the procedure to solve numerically the problem:

$r(0) = 0, \qquad r(1) = 2, \qquad r(2) = 5, \qquad r(3) = 9, \qquad r(4) = 12, \qquad r(5) = 14,$
$r(6) = 16, \qquad r(7) = r(8) = r(9) = r(10) = 17.$

For X units of resource and 2^N activities, how many additions are needed for solution?

Problem 3.4. Suppose there are two types of resources to be allocated to N activities with X units of resource-type 1 and Y units of resource-type 2. The return data form a set of N functions $r_k(x_k, y_k)$ giving the return from an allocation of x_k units of resource-type 1 and y_k units of resource-type 2 to activity k. Write the dynamic-programming formulation and determine the approximate number of additions as a function of X, Y, and N.

Problem 3.5. Use the procedure of Problem 3.4 to solve numerically the problem specified by the data given in Table 3.2, where $X = 3$ and $Y = 3$.

Table 3.2. *Data for Problem 3.5*

$r_1(x, y)$					$r_2(x, y)$					$r_3(x, y)$			
x \ y	0	1	2	3	x \ y	0	1	2	3	x \ y	0	1	2	3
0	0	1	3	6	0	0	2	4	6	0	0	3	5	8
1	4	5	6	7	1	1	4	6	7	1	2	5	7	9
2	5	6	7	8	2	4	6	8	9	2	4	7	9	11
3	6	7	8	9	3	6	8	10	11	3	6	9	11	13

The solution of Problem 3.4 involves computing a sequence of functions of two variables, one such function for each stage. We call this a two-dimensional problem. Henceforth, by the *dimension* of a problem we mean the number of state variables necessary for its dynamic-programming solution.

Problem 3.6. Suppose there are three types of resource, with initial quantities X, Y, and Z and return data $r_k(x_k, y_k, z_k)$, $k = 1, \ldots, N$. Give the dynamic-programming formulation and the approximate number of additions (in terms of X, Y, Z, and N) needed for solution. What is the dimension of this problem?

Problem 3.7. Do the problem of Section 1 subject to the additional constraint that the total allocation to a certain specified M of the N activities $(M < N)$ must be at least Y (Y specified, $Y < X$). Roughly how many additions are required for a dynamic-programming solution?

Problem 3.8. Do the problem of Section 1 with the additional constraint that at most M ($M < N$ and specified) of the allocations may be nonzero.

Problem 3.9. Consider the problem of Section 1. Suppose that if k of the N activities $(k = 1, \ldots, N)$ are used at a nonzero level, a cost of $h(k)$ is assessed against the total return. Give an efficient dynamic-programming solution and compare the number of additions to $NX^2/2$, the approximate number if the $h(k)$ cost is omitted.

Problem 3.10. A company has N divisions. An allocation of x units of resource to division k produces a dollar return $r_k(x)$ and also a return in terms of public goodwill $g_k(x)$. (Some divisions may be very profitable but create negative goodwill, while others may produce a good public image but little or no profit.) How would you use dynamic programming to maximize the dollar return to the company, given X units to allocate (but it need not all be used) subject to the constraint that total goodwill be $\geqslant$ (Y specified)? Approximately how many additions are required?

5. Unspecified Initial Resources

We now consider a problem slightly more complex than Problem 3.4, but show that its numerical solution by dynamic programming is actually somewhat easier.

Suppose that each unit of resource 1 costs $2 and each unit of resource 2 costs $3. We start with $Z = 2X + 3Y$ dollars and ask, "How many units of resource 1 and how many of resource 2 should we buy so that, if we then allocate them optimally to our N activities, we maximize the total return?" One decision is to buy X units of resource 1 and Y units of resource 2, getting the return obtained by the solution of Problem 3.4,

but we might do better by starting in some other initial resource situation. The data now must give $r_k(x_k, y_k)$ for all x_k and y_k such that $2x_k + 3y_k \leqslant Z$. Using the method of Problem 3.4 we could compute $f_1(x, y)$ for all x and y such that $2x + 3y \leqslant Z$ and then maximize $f_1(x, y)$ over all pairs x, y such that $2x + 3y \leqslant Z$. But we can do better.

First we compute for each activity a table $R_i(z)$, $z = 0, 1, \ldots, Z$, giving the optimal return that can be obtained from an allocation of z dollars. We do this by direct enumeration, maximizing over the number of units of the more expensive resource, yielding the formula

$$R_i(z) = \max_{y_i = 0, 1, \ldots, [z/3]} \left\{ r_i \left(\left[\frac{z - 3y_i}{2} \right], y_i \right) \right\}, \tag{3.6}$$

where $[a]$ means "the largest integer less than or equal to a." Hence $[z/3]$ is the greatest number of units of resource 2 that we can buy with z dollars and $[(z - 3y_i)/2]$ is the greatest number of units of resource 1 that we can buy if we have z dollars and buy y_i units of resource 2.

Once the N tables $R_i(z)$, $i = 1, \ldots, N$, have been computed, we define

$f_k(z) =$ the maximum return obtainable from allocating a
total of z dollars to activities k through N. $\qquad$ (3.7)

We can then write the usual recurrence relation

$$f_k(z) = \max_{z_k = 0, \ldots, z} \left[R_k(z_k) + f_{k+1}(z - z_k) \right] \tag{3.8}$$

and boundary condition

$$f_N(z) = R_N(z). \tag{3.9}$$

Computation of the number of additions and comparisons required by this method gives roughly $NZ^2/2$ of each for solution of (3.8), using the result given earlier in this chapter with Z replacing X, and roughly $NZ^2/6$ comparisons to determine the $R_i(z)$. The total is roughly NZ^2 operations. If this problem were solved by the standard method of Problem 3.4, $(N - 1)X^2Y^2/2$ operations would be required to compute f_1 for all values of resource 1 up to X and all values of resource 2 up to Y. Therefore, if $X = Z/2$ and $Y = Z/3$, then

$$\frac{(N - 1)(Z/2)^2(Z/3)^2}{2} = \frac{(N - 1)Z^4}{72} \approx \frac{NZ^4}{72}$$

operations would be required. In computing the latter formula we have neglected the fact that only x, y pairs such that $2x + 3y \leqslant Z$ need be considered, so the correct formula has a number larger than 72 in the denominator. However, the significant point is that one method grows like Z^2 and the other like Z^4, a very large difference if Z is, say, 100 or 1000.

Problem 3.11. Use the efficient method involving only one state variable to solve numerically the problem: choose nonnegative integers x_i and y_i, $i = 1, \ldots, 4$,

Table 3.3. *Data for Problem 3.11*

$r_1(x, y)$								$r_2(x, y)$						
x \ y	0	1	2	3	4	5	6	x \ y	0	1	2	3	4	5	6
0	0	5	9	11	12	13	14	0	0	3	5	6	7	8	9
1	5	9	13	18	×	×	×	1	2	5	9	12	×	×	×
2	10	×	×	×	×	×	×	2	7	×	×	×	×	×	×

$r_3(x, y)$								$r_4(x, y)$						
x \ y	0	1	2	3	4	5	6	x \ y	0	1	2	3	4	5	6
0	0	1	3	6	10	12	13	0	0	1	3	6	10	13	14
1	4	6	8	11	×	×	×	1	7	10	14	17	×	×	×
2	11	×	×	×	×	×	×	2	13	×	×	×	×	×	×

that maximize $\sum_{i=1}^{4} r_i(x_i, y_i)$ subject to $3\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i \leqslant 6$ for the data given in Table 3.3.

Problem 3.12. Write a one-dimensional recurrence relation for Problem 3.6 where we start with $\$W$ and one unit of resource 1 costs $\$a$, one unit of resource 2 costs $\$b$, and one unit of resource 3 costs $\$c$. For $a = b = c = 1$, how many additions and comparisons are needed for solution?

6. Lagrange Multipliers

In this section we shall see how Lagrange multipliers can be used to reduce the dimension of a dynamic-programming formulation. We shall demonstrate this computationally important procedure by considering a two-dimensional resource allocation problem of the type assigned in Problem 3.4. However, the procedure actually has much greater applicability. Some other applications appear in the problems.

Recall that the two-dimensional resource allocation problem can be stated mathematically as

$$\max\left[\sum_{i=1}^{N} r_i(x_i, y_i) \right]$$

$$\text{subject to} \quad \sum_{i=1}^{N} x_i = X,$$

$$\sum_{i=1}^{N} y_i = Y, \tag{3.10}$$

$$x_i = 0, 1, \ldots, X \quad (i = 1, 2, \ldots, N),$$

$$y_i = 0, 1, \ldots, Y \quad (i = 1, 2, \ldots, N).$$

Let us call this problem P.

In order to solve this problem by the Lagrange multiplier procedure, the following four steps are employed:

1. Guess a $\lambda \geqslant 0$.

2. Use dynamic programming to solve the following one-dimensional problem:

$$\max\left[\sum_{i=1}^{N} r_i(x_i, y_i) - \lambda \sum_{i=1}^{N} y_i \right]$$

subject to $\sum_{i=1}^{N} x_i = X,$

$$x_i = 0, 1, \ldots, X \quad (i = 1, 2, \ldots, N),$$

$$y_i = 0, 1, \ldots, Y \quad (i = 1, 2, \ldots, N).$$

Let us call this problem $\bar{P}(\lambda)$. If we solve $\bar{P}(\lambda)$ for a given λ, we get an optimal set of x_k and y_k. However, the solution to $\bar{P}(\lambda)$ may not be unique. Suppose that there are m distinct solutions. Let us denote these by $\{x_k^1(\lambda), y_k^1(\lambda)\}, \ldots, \{x_k^m(\lambda), y_k^m(\lambda)\}$.

3. Compute

$$F(\lambda) = \min_{j=1,\ldots,m} \sum_{i=1}^{N} y_i^j(\lambda) \qquad \text{and} \qquad G(\lambda) = \max_{j=1,\ldots,m} \sum_{i=1}^{N} y_i^j(\lambda).$$

4. (a) If $\sum_{i=1}^{N} y_i^j = Y$ for some j, then the original problem P is solved (see Theorem 3.1 of Section 7).

(b) If $G(\lambda) < Y$, then decrease λ and go to step 2 (see Theorem 3.2).

(c) If $F(\lambda) > Y$, then increase λ and go to step 2 (see Theorem 3.2).

(d) If $F(\lambda) < Y < G(\lambda)$ and for every j, $\sum_{i=1}^{N} y_i^j \neq Y$, then stop. The Lagrange multiplier method fails (see discussion of the gap problem in Section 7).

We now discuss step 2 in more detail. If $f_k(x; \lambda)$ is the optimal value function for $\bar{P}(\lambda)$, then the appropriate recurrence relation is

$$f_k(x; \lambda) = \max_{\substack{x_k=0, 1, \ldots, x \\ y_k=0, 1, \ldots, Y}} \left[r_k(x_k, y_k) - \lambda y_k + f_{k+1}(x - x_k; \lambda) \right]. \quad (3.11)$$

Notice that $f_{k+1}(x - x_k; \lambda)$ does not depend on y_k. Thus, since $f_k(x; \lambda)$ must be computed for different values of x, it is advantageous for us to maximize $r_k(x_k, y_k) - \lambda y_k$ as a function of y_k. Let

$$r_k'(x_k) = \max_{y_k=0, 1, \ldots, Y} \left[r_k(x_k, y_k) - \lambda y_k \right].$$

(A similar idea appeared in Section 5 where $R_i(z)$ was computed before dynamic programming was used.) Furthermore, keep track of the maximizing y_k for each value of x_k. We can rewrite (3.11) as

$$f_k(x; \lambda) = \max_{x_k=0, 1, \ldots, x} \left[r_k'(x_k) + f_{k+1}(x - x_k; \lambda) \right]. \quad (3.12)$$

This recurrence relation is equivalent to the one we used for the one-dimensional problem.

Problem 3.13. Use a Lagrange multiplier to replace the constraint on $\sum y_i$ in Problem 3.5. Solve for $\lambda = \frac{3}{2}$. For what value of Y have you now solved Problem 3.5?

7. Justification of the Procedure

In this section we shall justify the procedure that we have outlined above.

Theorem 3.1. If $\{x_i(\lambda), y_i(\lambda)\}$ is an optimal solution to $\bar{P}(\lambda)$, then it is also an optimal solution to P with Y replaced by $\sum_{i=1}^{N} y_i(\lambda)$.

Proof: A feasible solution to P is one satisfying the constraints (3.10). Let $\{\bar{x}_i, \bar{y}_i\}$ be any feasible solution to P with Y replaced by $\sum_{i=1}^{N} y_i(\lambda)$. Clearly, $\{\bar{x}_i, \bar{y}_i\}$ is also a feasible solution to $\bar{P}(\lambda)$. Therefore, by the optimality of $\{x_i(\lambda), y_i(\lambda)\}$ we get

$$\sum_{i=1}^{N} r_i[x_i(\lambda), y_i(\lambda)] - \lambda \sum_{i=1}^{N} y_i(\lambda) \geqslant \sum_{i=1}^{N} r_i(\bar{x}_i, \bar{y}_i) - \lambda \sum_{i=1}^{N} \bar{y}_i.$$

Since $\sum_{i=1}^{N} \bar{y}_i = \sum_{i=1}^{N} y_i(\lambda)$, we get

$$\sum_{i=1}^{N} r_i[x_i(\lambda), y_i(\lambda)] \geqslant \sum_{i=1}^{N} r_i(\bar{x}_i, \bar{y}_i).$$

Therefore, $\{x_i(\lambda), y_i(\lambda)\}$ is optimal for P with Y replaced by $\sum_{i=1}^{N} y_i(\lambda)$ and, thus, step 4a of the above procedure is justified. ∎

(Henceforth we use the symbol ∎ to indicate the conclusion of a proof.)

Theorem 3.2. If $\lambda_1 > \lambda_2$, then $G(\lambda_1) \leqslant F(\lambda_2)$.

Proof: Let $\{x_i^1, y_i^1\} = \{x_i^1(\lambda_1), y_i^1(\lambda_1)\}$ be the optimal solution of $\bar{P}(\lambda_1)$ for which $G(\lambda_1) = \sum_{i=1}^{N} y_i^1$ and let $\{x_i^2, y_i^2\} = \{x_i^2(\lambda_2), y_i^2(\lambda_2)\}$ be the optimal solution of $\bar{P}(\lambda_2)$ for which $F(\lambda_2) = \sum_{i=1}^{N} y_i^2$. By the optimality of $\{x_i^1, y_i^1\}$ and $\{x_i^2, y_i^2\}$ for their respective problems we get

$$\sum_{i=1}^{N} r_i(x_i^1, y_i^1) - \lambda_1 \sum_{i=1}^{N} y_i^1 \geqslant \sum_{i=1}^{N} r_i(x_i^2, y_i^2) - \lambda_1 \sum_{i=1}^{N} y_i^2, \qquad (3.13)$$

$$\sum_{i=1}^{N} r_i(x_i^2, y_i^2) - \lambda_2 \sum_{i=1}^{N} y_i^2 \geqslant \sum_{i=1}^{N} r_i(x_i^1, y_i^1) - \lambda_2 \sum_{i=1}^{N} y_i^1. \qquad (3.14)$$

If we add (3.13) and (3.14), we get (after canceling like terms and

rearranging)

$$(\lambda_1 - \lambda_2) \sum_{i=1}^{N} y_i^2 \geqslant (\lambda_1 - \lambda_2) \sum_{i=1}^{N} y_i^1.$$

Since $\lambda_1 > \lambda_2$, we get

$$F(\lambda_2) = \sum_{i=1}^{N} y_i^2 \geqslant \sum_{i=1}^{N} y_i^1 = G(\lambda_1). \quad \blacksquare$$

Thus, steps 4b and 4c of the above procedure are justified.

It is possible to give an economic interpretation to Theorem 3.2. Suppose that we have an unlimited amount of resource y but that we attach a cost of λ to each unit of it that we use. We seek a cost so that we use exactly the amount we really have. It is natural to expect that if the cost per unit is increased, then we shall use less of the resource.

Let us discuss now step 4d of the procedure. We have nowhere assumed that there exists a λ such that $\sum_{i=1}^{N} y_i(\lambda) = Y$. Even though $\sum_{i=1}^{N} y_i(\lambda)$ is a nonincreasing function of λ, it may not be a continuous function of λ. If $F(\lambda) < Y < G(\lambda)$ and for every j, $\sum_{i=1}^{N} y_i^j(\lambda) \neq Y$, then it follows from Theorem 3.2 that there is no value of λ that will produce an allocation of Y. Such values of Y are said to lie in a gap. How such gaps can occur will be illustrated in the next section. The existence of a gap does not mean that the original problem has no solution (in fact it does). What it does mean is that the procedure fails to find it. If the procedure fails, it may succeed if the roles of x and y are reversed (use λ to eliminate the x constraint and solve a one-dimensional dynamic-programming problem using the y constraint).

8. Geometric Interpretation of the Procedure

In this section we give a geometric interpretation of the Lagrange multiplier procedure by considering the following numerical example:

$$\max \sum_{i=1}^{2} r_i(x_i, y_i)$$

$$\text{subject to } \sum_{i=1}^{2} x_i = 3,$$

$$\sum_{i=1}^{2} y_i = 3,$$

$$x_i = 0, 1, 2, 3 \quad (i = 1, 2),$$

$$y_i = 0, 1, 2, 3 \quad (i = 1, 2),$$

Table 3.4.   *Data for example problem of Section 8*

	$r_1(x_1, y_1)$					$r_2(x_2, y_2)$			
x_1 \ y_1	0	1	2	3	x_2 \ y_2	0	1	2	3
0	0	1	5	6	0	0	3	5	6
1	1	3	6	7	1	2	4	7	8
2	3	5	7	8	2	3	5	8	9
3	4	6	8	10	3	4	6	9	10

where $r_i(x_i, y_i)$ for $i = 1, 2$, are given in Table 3.4. We shall attempt to solve this problem by using a Lagrange multiplier to eliminate the y constraint.

Consider all choices of x_1, y_1, x_2, and y_2 such that $x_1 + x_2 = 3$, $y_1 \leqslant 3$, $y_2 \leqslant 3$. Compute for each choice (there are 64 such choices) the resulting values of $r_1(x_1, y_1) + r_2(x_2, y_2)$ and $y_1 + y_2$. These values are plotted in Figure 3.2. There are fewer than 64 points because two different sets of x_1, y_1, x_2, and y_2 can give the same values of $r_1(x_1, y_1) + r_2(x_2, y_2)$ and $y_1 + y_2$. It is easy to see that the answer to our problem ($y_1 + y_2 = 3$) is 12.

Let us denote $r_1(x_1, y_1) + r_2(x_2, y_2)$ by v and $y_1 + y_2$ by u. We have a (u, v) coordinate system. For fixed c and λ, the curve $v = \lambda u + c$ is a straight line with v intercept c and slope λ. For fixed λ, finding x_1, y_1, x_2,
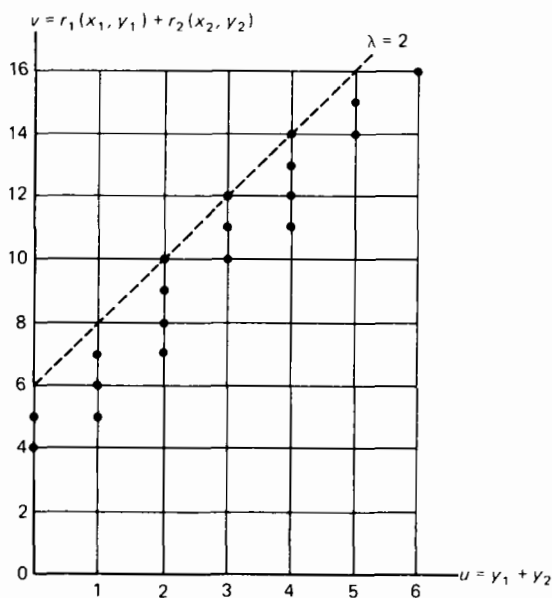
Figure 3.2

and y_2 (such that $x_1 + x_2 = 3$, $y_1 \leqslant 3$, and $y_2 \leqslant 3$) that maximizes c is the same as sliding the line upward (keeping the slope fixed) until it has the largest possible intercept while passing through a point of the plot. However,

$$c = v - \lambda u = \sum_{i=1}^{2} r_i(x_i, y_i) - \lambda \sum_{i=1}^{2} y_i.$$

Hence, solving $\bar{P}(\lambda)$ for a fixed λ is like sliding a line of slope λ up until it last touches a point of the plot. We want to choose λ so that the last point touched satisfies $\sum_{i=1}^{2} y_i = 3$. We can do this by choosing $\lambda = 2$ (see Figure 3.2). However, the solution is not unique. We get $\sum_{i=1}^{2} y_i = 2$, 3, or 4.

In Figure 3.3 we have plotted $\sum_{i=1}^{2} y_i(\lambda)$ as a function of λ. Notice that $\sum_{i=1}^{2} y_i = 1$ lies in a gap, i.e., the procedure fails for $\sum_{i=1}^{2} y_i = 1$. This can also be seen from Figure 3.2. If we consider the graph of the optimal value of $\sum_{i=1}^{2} r_i(x_i, y_i)$ as a function of $\sum_{i=1}^{2} y_i$, then it can be seen that $\sum_{i=1}^{2} y_i = 1$ lies in a region of nonconcavity of this graph.

Let us summarize our procedure. The Lagrange multiplier procedure seeks a slope λ such that maximizing $\sum_{i=1}^{N} r_i(x_i, y_i) - \lambda \sum_{i=1}^{N} y_i$ gives the highest point of the plot (of admissible x_i and y_i) on the vertical line $\sum_{i=1}^{N} y_i = Y$ (i.e., the solution of the original problem).
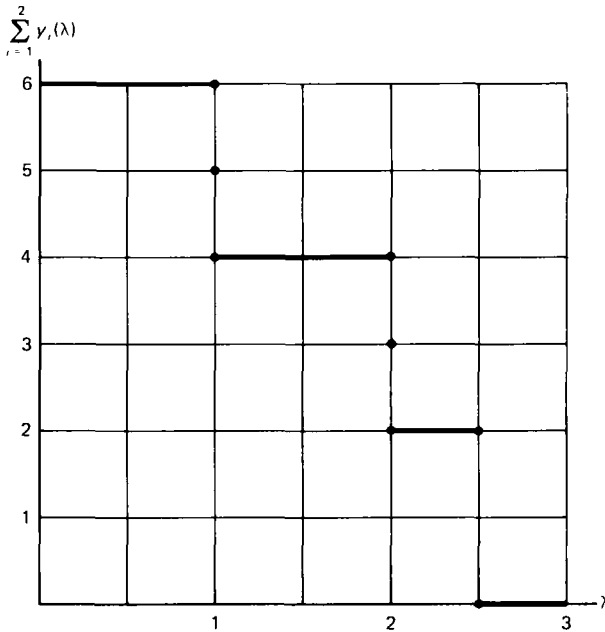
Figure 3.3
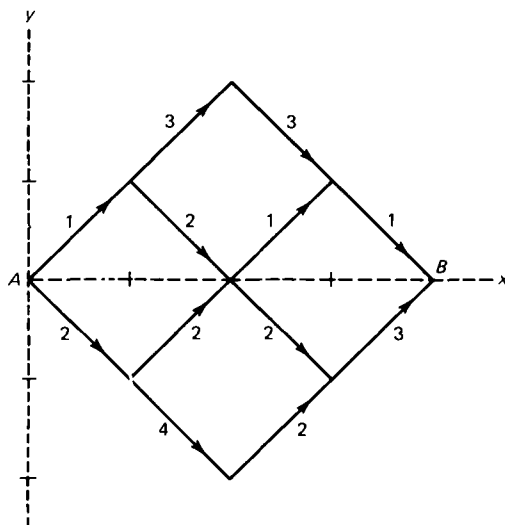
Figure 3.4

Problem 3.14. On the network given in Figure 3.4, problem P seeks the path from A to B with minimum sum of arc costs subject to the additional restriction that the path must have exactly two direction changes. Plot the cost v as a function of the number of turns u for all six paths between A and B. Find from the plot what cost λ, if such a λ exists, we can charge for each direction change in a new problem $\bar{P}(\lambda)$ that seeks the cheapest path including this turn cost while ignoring the two-turn constraint, such that the solution of $\bar{P}$ solves problem P. Solve $\bar{P}$ for one such λ (it may not be unique) by the methods of Chapter 1. Verify that the solution path has exactly two direction changes.

Problem 3.15. Consider the numerical example of Section 8; in particular, see Figure 3.2. For a given λ, slide a straight line of slope λ up until it last touches a point of the plot. Let $f(\lambda)$ be the height of this line above $y_1 + y_2 = 3$. (The dashed line in Figure 3.2 gives $f(2) = 12$.) Plot $f(\lambda)$ for $\lambda = 0$, $\frac{1}{2}$, 1, $1\frac{1}{2}$, 2, $2\frac{1}{2}$, and 3 and then connect these points by straight line segments ($f(\lambda)$ is a piecewise-linear function). Notice that the minimum of $f(\lambda)$ occurs at $\lambda = 2$, the same value of λ that solved the numerical example. The significance of this remark will be made clear in the solution to this problem.
Repeat the above for the height above $y_1 + y_2 = 1$.

9. Some Additional Cases

Note from Figure 3.2 that the maximum v is a nondecreasing function of u since the return functions are assumed nondecreasing in their arguments. Hence the appropriate slope λ, if one exists, is nonnegative.
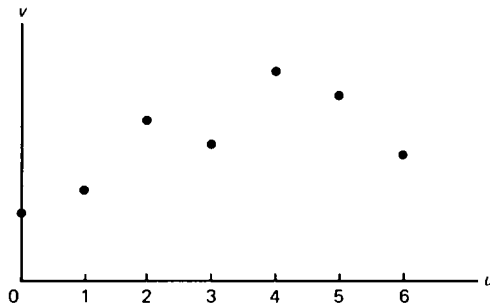
Figure 3.5

If the r_i were not nondecreasing and the constraint on $\sum_{i=1}^{N} y_i$ were an equality constraint, the maximum value of v as a function of u could look as in Figure 3.5. Then all appropriate slopes λ for $Y = 5$ and $Y = 6$ would be negative while they would be positive for $Y = 0$ and 2. No λ exists for $Y = 1$ and $Y = 3$ (gaps exist), and there are both positive and negative λ's that are appropriate when $Y = 4$. For general return data, step 1 of the procedure of Section 6 can put no constraint on the sign of λ.

Suppose that the r_i are not necessarily nondecreasing and that the constraint to be eliminated by the Lagrange multuplier procedure is the *inequality* constraint

$$\sum_{i=1}^{N} y_i \leqslant Y. \tag{3.15}$$

First, we would solve the problem ignoring the constraint, i.e., letting $\lambda = 0$ and using the procedure of Section 6. If a resulting optimal policy satisfies (3.15), the original problem has been solved.

If the resulting policies violate (3.15), we would solve with equality required in (3.15). We would restrict ourselves to $\lambda > 0$. If such a λ can be found, the original inequality-constrained problem has been solved.

Problem 3.16. Why?

The problem with equality constraint cannot be solved with a negative λ if the solution for $\lambda = 0$ violates (3.15).

If the method fails because no λ yields a solution to the equality-constrained problem (a gap), it may be because the optimal solution satisfying (3.15) yields *equality* in the constraint but lies in a gap (such is the case for the constraint $u \leqslant 1$ in Figure 3.5) or because the optimal solution satisfying (3.15) yields *strict inequality* in (3.15) and is therefore a relative maximum but the global maximum violates the constraint (such is the case for $u \leqslant 3$ in Figure 3.5). Hence, if we encounter

a gap, we cannot even say whether the optimal solution satisfies (3.15) as an equality or a strict inequality.

10. More Than Two Constraints

The Lagrange multiplier procedure can be extended to problems with more than two constraints. Let problem P now be as follows:

$$\max \sum_{i=1}^{N} r_i(x_{i1}, x_{i2}, \ldots, x_{il})$$

$$\text{subject to } \sum_{i=1}^{N} x_{ij} = X_j \quad (j = 1, 2, \ldots, l),$$

$$x_{ij} = 0, 1, \ldots, X_j \quad \text{(for all } i \text{ and } j\text{)}.$$

Suppose we introduce k $(k < l)$ Lagrange multipliers. Problem $\bar{P}(\lambda_1, \lambda_2, \ldots, \lambda_k)$ is now as follows:

$$\max \left[\sum_{i=1}^{N} r_i(x_{i1}, x_{i2}, \ldots, x_{il}) - \sum_{j=1}^{k} \lambda_j \sum_{i=1}^{N} x_{ij} \right]$$

$$\text{subject to } \sum_{i=1}^{N} x_{ij} = X_j \quad (j = k+1, \ldots, l),$$

$$x_{ij} = 0, 1, \ldots, X_j \quad \text{(for all } i \text{ and } j\text{)}.$$

For each set of multipliers $\lambda_1, \lambda_2, \ldots, \lambda_k$ we must solve an $(l - k)$-dimensional dynamic-programming problem. The smaller k is, the more computation is necessary for a given set of multipliers, the fewer λ_i to search for, and the less the likelihood of a gap causing failure of the procedure. The best choice of k is very problem-dependent and also depends on such factors as the size of the computer memory that is available.

Problem 3.17. Consider the problem: choose nonnegative integers x_i, $i = 0, \ldots, N$, that maximize J given by

$$J = \sum_{i=0}^{N-1} g_i(x_i, x_{i+1})$$

$$\text{subject to } \sum_{i=0}^{N} x_i = X \quad (X \text{ given}).$$

Give a dynamic-programming solution. How would you use a Lagrange multiplier to try to reduce the computation?

Problem 3.18. In Problem 3.10, how would you use the Lagrange multiplier method to try to solve the problem? Do not assume that goodwill is a

nondecreasing function of x. Approximately how many additions are required for one fixed value of λ? What is the sign of λ?

Problem 3.19. Consider the problem: choose nonnegative integers x_i, y_i, $i = 1, \ldots, N$, that maximize J given by

$$J = \sum_{i=1}^{2^N} r(x_i, y_i)$$

$$\text{subject to } \sum_{i=1}^{2^N} x_i = X,$$

$$\sum_{i=1}^{2^N} y_i = Y.$$

Note that the return function $r(x, y)$ is the same for all i. Give an efficient dynamic-programming solution using a Lagrange multiplier and "doubling-up" the number of stages at each iteration. For a fixed value of λ and for $N = 5$, $X = 1000$, and $Y = 1000$, approximately how many additions are required for solution?

Chapter 4

THE GENERAL SHORTEST-PATH PROBLEM

1. Introduction

Consider a network that has a set of nodes N (these nodes are arbitrarily numbered from 1 to N) and a set of arcs A. We shall denote the directed arc from node i to node j by (i, j). Every arc $(i, j) \in A$ (this notation means (i, j) is an element of the set A) has associated with it a distance (or travel time, or cost, etc.) d_{ij}. We shall assume (except in Section 2) that every pair of nodes is connected by a directed arc; however, some arcs may be infinite in length. Assume further that $d_{ii} = 0$. We shall not, however, assume that $d_{ij} = d_{ji}$ or that the d_{ij} satisfy the triangle inequality, $d_{ij} + d_{jk} \geqslant d_{ik}$.

In this chapter we shall present a general treatment of shortest-path problems. The basic shortest-path problem is as follows: in the network described above find the length of the shortest path from node 1 to node N. In Section 2 we consider acyclic networks, i.e., networks that contain no cycles. (A *cycle* is a path that begins and ends at the same node.) For such networks there is a natural stage variable that increases along any path. This allows a simple dynamic-programming formulation. In Section 3 we consider networks that may contain cycles. A natural stage variable no longer exists; however, by being sufficiently clever, it is still possible to give a dynamic-programming formulation for the shortest-path problem.

Shortest-path problems occur in many disciplines. For example, they have applications in such areas as transportation planning and communication routing. They also occur as subproblems in other problems of optimization (see, for example, Chapter 5).

2. Acyclic Networks

In this section we consider shortest-path problems for acyclic networks. It should be noted that all of the networks considered in Chapter 1 were acyclic. We accomplished this simplification by considering only networks all of whose arcs were directed to the right.

In an acyclic network it is possible to find an enumeration of the nodes such that if $(i, j) \in A$, then $i < j$. To accomplish such an enumeration, designate as node 1 a node that has only outward-pointing arcs. (If there is more than one such node, then any one of the nodes may be chosen.)

Problem 4.1. Show that every acyclic network has at least one node that has only outward-pointing arcs.

Check off this node (which has only outward-pointing arcs) and also all of its arcs, and consider them no further. The network that remains is also acyclic. Therefore, it has a node that has only outward-pointing arcs. Designate this as node 2. Check off this node and all of its arcs, and consider them no further. Continue in this manner until all nodes have been enumerated.

Problem 4.2. For the acyclic network shown in Figure 4.1, enumerate the nodes such that if $(i, j) \in A$, then $i < j$.
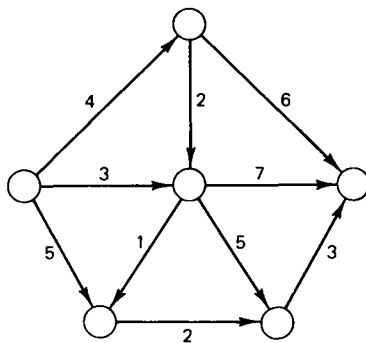

Figure 4.1

Suppose we have an acyclic network that has been enumerated as above. We now present the (forward) dynamic-programming formulation for the problem of finding the shortest distance from node 1 to node N. Actually, we shall solve the more general problem of finding the shortest distances from node 1 to all nodes. Define the optimal value function f_i as

follows:

$$f_i = \text{the length of the shortest path from node 1 to node } i. \quad (4.1)$$

Then the dynamic-programming formulation is as follows:

recurrence relation: $\quad f_i = \min_{j < i} [f_j + d_{ji}] \qquad (i = 2, 3, \ldots, N)$

$$(d_{ji} = \infty \quad \text{if} \quad (j, i) \notin A), \qquad (4.2)$$

boundary condition: $\quad f_1 = 0,$ $\qquad\qquad\qquad\qquad\qquad\qquad (4.3)$

answers: $\quad f_i \quad$ for $\quad i = 1, 2, \ldots, N.$ $\qquad\qquad\qquad\qquad (4.4)$

Notice that the optimal value function is a function only of a stage variable. This is because there is only one possible state at each stage.

We shall now show, using mathematical induction, that f_i is correctly given by (4.2) and (4.3). Consider first the case $i = 2$. The only path from node 1 to node 2 is via the directed arc (1, 2). Therefore, $f_2 = d_{12}$, as given by (4.2) and (4.3).

Assume now that f_j is correctly given by (4.2) and (4.3) for $j = 1, 2, \ldots, i - 1$. We must show that f_i is correctly given by (4.2). Every arc entering node i must come from a lower numbered node, say, node j. Furthermore, it follows from our inductive hypothesis that the length of the shortest path from node 1 to node i that has node j immediately preceding node i is given by $f_j + d_{ji}$. Since we are free to choose node j optimally, f_i is correctly given by (4.2).

Thus, the lengths of the shortest paths from node 1 to all other nodes can be obtained by recursively computing $f_2, f_3, \ldots, f_N$. If we let p_i be the optimal policy at node i (i.e., the node that immediately precedes node i on the shortest path from node 1 to node i), then p_i can be obtained by recording that node j that minimizes the right-hand side (r.h.s.) of (4.2).

Problem 4.3. Use (4.2)–(4.4) to find the shortest distances from node 1 to all nodes for the network given in Figure 4.1.

Problem 4.4. Give a backward dynamic-programming formulation for the problem of finding the shortest distances from all nodes to node N in an acyclic network.

We conclude our discussion of acyclic networks by calculating the number of elementary operations (i.e., additions and comparisons) required by the above dynamic-programming formulation. At stage 2, one addition and no comparisons are required. At stage 3, two additions and one comparison are required. Finally, at stage N, $N - 1$ additions and $N - 2$ comparisons are required. Therefore, for all stages $N(N - 1)/2$ additions and $(N - 1)(N - 2)/2$ comparisons, or approximately N^2

operations, are required. Note that these calculations do not include the effort needed to enumerate the nodes initially.

3. General Networks

In this section we consider shortest-path problems for a general network, i.e., a network that may contain cycles. There is no longer a natural ordering of the nodes and, thus, the dynamic-programming formulation of the preceding section is not applicable. We shall consider two different versions of the shortest-path problem: (A) determining the shortest path between a specified pair of nodes of a network, and (B) determining the shortest path between each pair of nodes of a network.

A. The Shortest Path between a Specified Pair of Nodes

We are given a network with N nodes, numbered arbitrarily from 1 to N. The problem is to find the shortest path from node 1 to node N. However, as is usual in dynamic programming, the procedures that we present will actually find the shortest paths from node 1 to all other nodes.

Let us assume initially that $d_{ij} \geqslant 0$ for every arc (i, j). The most efficient procedure for this problem was first described by Dijkstra [3]. Since the dynamic-programming formulation of Dijkstra's procedure is not obvious, we shall initially present it in the form of an algorithm.

At some step in the procedure a node, i, may have a temporary label, $\Pi(i)$, or a permanent label, $\overline{\Pi}(i)$. A temporary label is an upper bound on the shortest distance from node 1 to node i, while a permanent label is the true shortest distance from node 1 to node i. Let $N(1)$ be the set of nodes with permanent labels. Initially $N(1) = \{1\}$ and $\overline{\Pi}(1) = 0$. All other nodes have temporary labels, where $\Pi(i) = d_{1i}$. We shall use the symbol $:=$ to mean "is replaced by." Dijkstra's procedure is as follows:

Step 1: Determine a node with a temporary label, say node k, such that $\Pi(k) = \min_{i \notin N(1)} \Pi(i)$. (If there is more than one node k, then take the node with the smallest node number.) Set $\overline{\Pi}(k) = \Pi(k)$ and $N(1) := N(1) \cup \{k\}$.

Step 2: If $N(1) = N$, then stop. If not, go to step 3.

Step 3: For each arc (k, l) (k is the node determined in step 1) such that $l \notin N(1)$, $\Pi(l) := \min[\Pi(l), \overline{\Pi}(k) + d_{kl}]$. Go to step 1.

In order to determine the arcs that make up the shortest paths, a table can be constructed giving the node from which each permanently labeled node was labeled.

We shall now show that on each iteration of Dijkstra's procedure (i.e., steps 1–3) one true shortest distance is determined (an additional node is added to $N(1)$) and, furthermore, that for each node j not in $N(1)$ after step 3, $\Pi(j)$ represents the shortest distance from node 1 to node j that can be obtained by a path in which all but the terminal node belong to $N(1)$. We shall use mathematical induction; however, we shall give only the crucial inductive step. Suppose that we have determined $i - 1$ true shortest distances; that is, $N(1)$ now contains $i - 1$ nodes. Furthermore, assume that for each node j not in $N(1)$, $\Pi(j)$ has the above interpretation. We claim that the node, k, with the smallest temporary label can be added to $N(1)$. That is, $\Pi(k)$ is actually the true shortest distance from node 1 to node k; because if a shorter path from node 1 to node k exists, then it would have to contain a first node, j, that is currently not in $N(1)$. However, $\Pi(j) \geqslant \Pi(k)$ and the path from node j to node k must have nonnegative length since we have assumed that all distances are nonnegative. Therefore, $\overline{\Pi}(k) = \underline{\Pi}(k)$ and node k may be added to $N(1)$. Furthermore, we claim that after $\overline{\Pi}(k)$ has been used in step 3 to update the temporary label of each node, l, not in $N(1)$, $\Pi(l)$ once again represents the shortest distance from node 1 to node l that can be obtained by a path in which all but the terminal node belong to $N(1)$. The only type of path that could have been overlooked is one that includes node k, but the node preceding node l is some node in $N(1)$ other than k, say node m. However, the length of such a path can be no better than the previous value of $\Pi(l)$ ($\Pi(l) \leqslant \overline{\Pi}(m) + d_{ml}$) since node k is not on the shortest path from node 1 to node m. We have therefore completed the inductive step.

Let us now apply Dijkstra's procedure to the network given in Figure 4.2. The calculations for this problem are as follows:

Initial conditions:

$$\overline{\Pi}(1) = 0, \quad \Pi(2) = 5, \quad \Pi(3) = 1, \quad \Pi(4) = \infty, \quad \Pi(5) = \infty, \quad \Pi(6) = \infty.$$

Iteration 1: Node 3 has the smallest temporary label and thus its label becomes permanent.
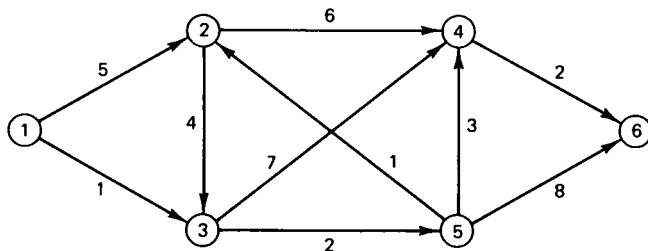
Figure 4.2

$$\overline{\Pi}(1) = 0,$$

$$\overline{\Pi}(3) = 1,$$

$$\Pi(2) := \min\left[\Pi(2), \overline{\Pi}(3) + d_{32}\right] = \min[5, 1 + \infty] = 5,$$

$$\Pi(4) := \min\left[\Pi(4), \overline{\Pi}(3) + d_{34}\right] = \min[\infty, 1 + 7] = 8,$$

$$\Pi(5) := \min\left[\Pi(5), \overline{\Pi}(3) + d_{35}\right] = \min[\infty, 1 + 2] = 3,$$

$$\Pi(6) := \min\left[\Pi(6), \overline{\Pi}(3) + d_{36}\right] = \min[\infty, 1 + \infty] = \infty.$$

Iteration 2: Node 5 has the smallest temporary label and thus its label becomes permanent.

$$\overline{\Pi}(1) = 0,$$

$$\overline{\Pi}(3) = 1,$$

$$\overline{\Pi}(5) = 3,$$

$$\Pi(2) := \min\left[\Pi(2), \overline{\Pi}(5) + d_{52}\right] = \min[5, 3 + 1] = 4,$$

$$\Pi(4) := \min\left[\Pi(4), \overline{\Pi}(5) + d_{54}\right] = \min[8, 3 + 3] = 6,$$

$$\Pi(6) := \min\left[\Pi(6), \overline{\Pi}(5) + d_{56}\right] = \min[\infty, 3 + 8] = 11.$$

Iteration 3: Node 2 has the smallest temporary label and thus its label becomes permanent.

$$\overline{\Pi}(1) = 0,$$

$$\overline{\Pi}(3) = 1,$$

$$\overline{\Pi}(5) = 3,$$

$$\overline{\Pi}(2) = 4,$$

$$\Pi(4) := \min\left[\Pi(4), \overline{\Pi}(2) + d_{24}\right] = \min[6, 4 + 6] = 6,$$

$$\Pi(6) := \min\left[\Pi(6), \overline{\Pi}(2) + d_{26}\right] = \min[11, 4 + \infty] = 11.$$

Iteration 4: Node 4 has the smallest temporary label and thus its label becomes permanent.

$$\overline{\Pi}(1) = 0,$$

$$\overline{\Pi}(3) = 1,$$

$$\overline{\Pi}(5) = 3,$$

$$\overline{\Pi}(2) = 4,$$

$$\overline{\Pi}(4) = 6,$$

$$\Pi(6) = \min\left[\Pi(6), \overline{\Pi}(4) + d_{46}\right] = \min[11, 6 + 2] = 8.$$

Iteration 5:

$$\overline{\Pi}(1) = 0, \quad \overline{\Pi}(3) = 1, \quad \overline{\Pi}(5) = 3, \quad \overline{\Pi}(2) = 4, \quad \overline{\Pi}(4) = 6, \quad \overline{\Pi}(6) = 8.$$

The length of the shortest path from node 1 to node i is given in the box next to node i in Figure 4.3. The shortest paths are indicated by heavy lines.

Let us compute the number of operations required by Dijkstra's procedure. In order to compute the shortest paths from node 1 to all other nodes, a total of $N - 1$ iterations are required. At iteration i, $(N - 1) - i$ comparisons are required to determine the minimum temporary label, and $(N - 1) - i$ additions and $(N - 1) - i$ comparisons are required to update the temporary labels. Therefore, for all iterations $(N - 1)(N - 2)/2$ additions and $(N - 1)(N - 2)$ comparisons, or approximately $3N^2/2$ operations, are required.

In computing the number of operations required by Dijkstra's procedure, we did not consider the problem of how to distinguish between temporarily and permanently labeled nodes. However, Yen [10] has shown that by combining Dijkstra's procedure with a simple list-processing technique, no more additions or comparisons are required to find all shortest paths than the number computed above. Therefore, when properly implemented, Dijkstra's procedure can solve the single-pair problem with nonnegative distances in approximately $3N^2/2$ operations. It should be noted, however, that there is a minor error in Yen's paper which was found and corrected by Williams and White [8].

We now show that Dijkstra's procedure can be given a dynamic-programming formulation. Let $N_i(1)$ be the set composed of the i closest nodes (by shortest path) to node 1, where $N_1(1) = \{1\}$. (In case this set is not well defined due to ties, $N_i(1)$ is a set of i nodes such that no node not in $N_i(1)$ is closer to 1 than some node in $N_i(1)$.) Define the optimal value function $f_i(j)$ as follows:
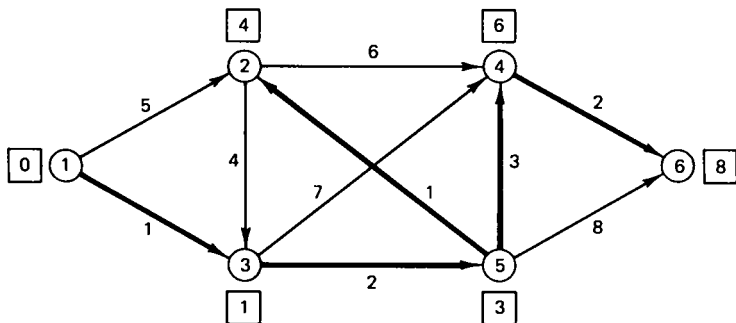
Figure 4.3

$f_i(j) =$ the length of the shortest path from node 1 to node j when we are restricted to use paths such that all nodes preceding node j belong to $N_i(1)$.   (4.5)

Then the dynamic-programming formulation is as follows:

At stage i $(i = 2, 3, \ldots, N)$ determine a node, k_i, such that $k_i \notin N_{i-1}(1)$ and $f_{i-1}(k_i) = \min_{j \notin N_{i-1}(1)} f_{i-1}(j)$. Set $N_i(1) = N_{i-1}(1) \cup \{k_i\}$. Then compute $f_i(j)$ from

recurrence relation:

$$f_i(j) = \begin{bmatrix} f_{i-1}(j) & \text{if } j \in N_i(1), \\ \min\{f_{i-1}(j), f_{i-1}(k_i) + d_{k_i, j}\} & \text{if } j \notin N_i(1); \end{bmatrix} \qquad (4.6)$$

boundary conditions: $N_1(1) = \{1\}$, $k_1 = 1$, and $f_1(j) = d_{1j}$;   (4.7)

answers: $f_i(k_i)$ for $i = 1, 2, \ldots, N$.   (4.8)

It is easy to see that this dynamic-programming formulation is essentially equivalent to the algorithm presented above.

Dijkstra's procedure, which we considered above, required $d_{ij} \geqslant 0$ for every arc (i, j). Suppose now that we would like to find the shortest distances from node 1 to all other nodes in a general network which has one or more negative d_{ij}. Such a problem may arise when arc numbers represent costs, and some arcs are profitable. Since this shortest-path problem will have an unbounded solution if a negative cycle (a cycle for which the sum of the arc numbers is negative) exists, we would like a solution procedure that can detect such cycles. We shall present three successively more efficient procedures for solving the described problem.

Procedure 1 was originally proposed for problems with $d_{ij} \geqslant 0$ by several authors (see, for example, Ford [5]). Define the optimal value function $f_i(k)$ (for $i = 1, 2, \ldots$) as follows:

$f_i(k) =$ the length of the shortest path from node 1 to node k when i or fewer arcs must be used.   (4.9)

Then the dynamic-programming formulation is as follows:

recurrence relation:

$$f_i(k) = \min_{j \neq k} \left[f_{i-1}(j) + d_{jk} \right] (i = 1, 2, \ldots, N; \quad k = 1, 2, \ldots, N) \quad (4.10)$$

(except that for $k = 1$, include $j = k$ in the minimization of the r.h.s.);

boundary conditions:

$$f_0(k) = \begin{bmatrix} 0 & \text{if } k = 1, \\ \infty & \text{if } k = 2, 3, \ldots, N; \end{bmatrix} \qquad (4.11)$$

answers: $f_N(k)$ for $k = 1, 2, \ldots, N$.   (4.12)

Let us show that $f_i(k)$ is correctly given by (4.10) and (4.11). We shall use mathematical induction. The case $i = 1$ is obvious. Assume that $f_{i-1}(k)$ is correctly given by (4.10) for all k. We must show the same for $f_i(k)$. Suppose that j is the node immediately preceding node k on the shortest path from node 1 to node k using i or fewer arcs. It is clear that such a path must go optimally from node 1 to node j in $i - 1$ or fewer arcs. Therefore, by our inductive hypothesis the length of such a path is $f_{i-1}(j)$ $+ d_{jk}$. Since we are free to choose j optimally from the set $N - \{k\}$, the correctness of (4.10) is established.

A word of explanation is necessary for (4.12). If $f_i(k) = f_{i-1}(k)$ $(k = 1, 2, \ldots, N)$ for some i, $2 \leqslant i \leqslant N$ (this is called *convergence*), then stop. The answer is $f_i(k)$ since no improvement is possible. (If $f_j(k) = f_{i-1}(k)$ $(k = 1, 2, \ldots, N)$ for $j \geqslant i$, then a negative cycle cannot exist.) If convergence does not occur by the Nth iteration, then a negative cycle exists and the problem has an unbounded solution. (If no negative cycles exist, then no shortest path can contain more than $N - 1$ arcs.)

The node that precedes node k on the shortest path from node 1 to node k when i or fewer arcs must be used, call it $p_i(k)$, can be determined by recording that node, j, which minimizes the r.h.s. of (4.10).
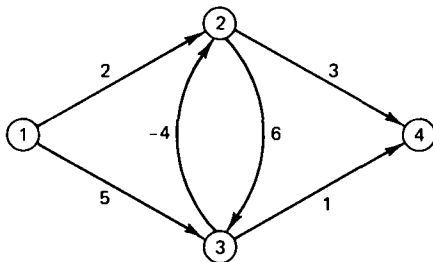
Figure 4.4

We now apply the above procedure to the network given in Figure 4.4. The calculations are as follows:

$$f_0(1) = 0,$$
$$f_0(2) = \infty,$$
$$f_0(3) = \infty,$$
$$f_0(4) = \infty.$$

$$f_1(1) = \min[0 + 0, \infty + \infty, \infty + \infty, \infty + \infty] = 0, \qquad p_1(1) = 1;$$
$$f_1(2) = \min[0 + 2, \infty - 4, \infty + \infty] = 2, \qquad p_1(2) = 1;$$
$$f_1(3) = \min[0 + 5, \infty + 6, \infty + \infty] = 5, \qquad p_1(3) = 1;$$
$$f_1(4) = \min[0 + \infty, \infty + 3, \infty + 1] = \infty, \qquad p_1(4) = 1, 2, \text{ or } 3.$$

$$f_2(1) = \min[\,0 + 0,\ 2 + \infty,\ 5 + \infty,\ \infty + \infty\,] = 0, \qquad p_2(1) = 1;$$
$$f_2(2) = \min[\,0 + 2,\ 5 - 4,\ \infty + \infty\,] = 1, \qquad p_2(2) = 3;$$
$$f_2(3) = \min[\,0 + 5,\ 2 + 6,\ \infty + \infty\,] = 5, \qquad p_2(3) = 1;$$
$$f_2(4) = \min[\,0 + \infty,\ 2 + 3,\ 5 + 1\,] = 5, \qquad p_2(4) = 2.$$

$$f_3(1) = \min[\,0 + 0,\ 1 + \infty,\ 5 + \infty,\ 5 + \infty\,] = 0, \qquad p_3(1) = 1;$$
$$f_3(2) = \min[\,0 + 2,\ 5 - 4,\ 5 + \infty\,] = 1, \qquad p_3(2) = 3;$$
$$f_3(3) = \min[\,0 + 5,\ 1 + 6,\ 5 + \infty\,] = 5, \qquad p_3(3) = 1;$$
$$f_3(4) = \min[\,0 + \infty,\ 1 + 3,\ 5 + 1\,] = 4, \qquad p_3(4) = 2.$$

$$f_4(1) = \min[\,0 + 0,\ 1 + \infty,\ 5 + \infty,\ 4 + \infty\,] = 0, \qquad p_4(1) = 1;$$
$$f_4(2) = \min[\,0 + 2,\ 5 - 4,\ 4 + \infty\,] = 1, \qquad p_4(2) = 3;$$
$$f_4(3) = \min[\,0 + 5,\ 1 + 6,\ 4 + \infty\,] = 5, \qquad p_4(3) = 1;$$
$$f_4(4) = \min[\,0 + \infty,\ 1 + 3,\ 5 + 1\,] = 4, \qquad p_4(4) = 2.$$

Therefore, the shortest path from node 1 to node 4 is 1–3–2–4 ($p_4(4) = 2$, $p_3(2) = 3$, $p_2(3) = 1$) and its length is 4.

We now present procedure 2. At iteration i of procedure 1, we successively compute $f_i(1), f_i(2), \ldots, f_i(N)$ in terms of $f_{i-1}(j)$ ($j = 1, 2, \ldots, N$). Thus, when we compute $f_i(k)$ we already know $f_i(1), f_i(2), \ldots, f_i(k-1)$. Furthermore, $f_i(j) \leqslant f_{i-1}(j)$ for $j = 1, 2, \ldots, k-1$. It would therefore seem advantageous to use $f_i(j)$ ($j = 1, 2, \ldots, k-1$) rather than $f_{i-1}(j)$ ($j = 1, 2, \ldots, k-1$) to compute $f_i(k)$. Procedure 2 does exactly this. Let $1 - i_1 - i_2 - \cdots - i_m - k$ be a path from node 1 to node k. A *decrease* is said to occur at node i_p if $i_p < i_{p-1}$. For example, the path 1–5–2–4–3 has two decreases. Define the optimal value function $g_i(k)$ (for $i = 1, 2, \ldots$) as follows:

$g_i(k) = $ the length of the shortest path from node 1 to node k
   among all paths having $i - 1$ or fewer decreases.   (4.13)

Then the dynamic-programming formulation is as follows:

recurrence relation:

$$g_i(k) = \min \left[\begin{array}{c} \min\limits_{j<k} \{ g_i(j) + d_{jk} \} \\[2mm] \min\limits_{j>k} \{ g_{i-1}(j) + d_{jk} \} \end{array} \right] \qquad (i = 1, 2, \ldots, N; \quad k = 1, 2, \ldots, N)$$

(4.14)

(except that for $k = 1$, include $j = k$ in the bottom minimization on the r.h.s.);

boundary conditions:

$$g_0(k) = \begin{bmatrix} 0 & \text{if} & k = 1, \\ \infty & \text{if} & k = 2, 3, \ldots, N; \end{bmatrix} \tag{4.15}$$

answers: $g_N(k)$ for $k = 1, 2, \ldots, N$. $\tag{4.16}$

Furthermore, the stopping rule for procedure 2 is the same as for procedure 1.

Problem 4.5. Show that $g_i(k)$, as defined by (4.13), is correctly given by (4.14) and (4.15).

Problem 4.6. Find the shortest paths from node 1 to all nodes for the network given in Figure 4.4 using procedure 2.

Problem 4.7. Show that $g_i(k) \leqslant f_i(k)$ $(k = 1, 2, \ldots, N)$ for $i = 1, 2, \ldots$. Since procedures 1 and 2 require the same number of operations per iteration, it follows that the number of operations required by procedure 2 for any given problem is less than or equal to the number required by procedure 1 for the same problem.

Procedure 3, which we now present, was first proposed by Yen [9]. Both of the previous two procedures always process the nodes in a forward manner, i.e., $k = 1, 2, \ldots, N$. However, procedure 3 alternately processes the nodes first in a forward manner and then in a backward manner $(k = N, N - 1, \ldots, 1)$. Let $1 - i_1 - i_2 - \cdots - i_m - k$ be a path from node 1 to node k. A *reversal* is said to occur at node i_p if $i_p < i_{p-1}$ and $i_{p+1} > i_p$, or if $i_p > i_{p-1}$ and $i_{p+1} < i_p$. For example, the path $1-5-2-4-3$ has three reversals. Define the optimal value function $h_i(k)$ (for $i = 1, 2, \ldots$) as follows:

$h_i(k) =$ the length of the shortest path from node 1 to node k
　　　　among all paths having $i - 1$ or fewer reversals. $\tag{4.17}$

Then the dynamic-programming formulation is as follows:

recurrence relations:

$$h_i(k) = \min \begin{bmatrix} h_{i-1}(k) \\ \min_{j<k} \{ h_i(j) + d_{jk} \} \end{bmatrix} \quad (i = 1, 3, 5, \ldots; \quad k = 1, 2, \ldots, N),$$

$$\tag{4.18}$$

$$h_i(k) = \min \begin{bmatrix} h_{i-1}(k) \\ \min_{j>k} \{ h_i(j) + d_{jk} \} \end{bmatrix} \quad \begin{array}{l} (i = 2, 4, 6, \ldots; \\ k = N, N - 1, \ldots, 1); \end{array}$$

boundary conditions:

$$h_0(k) = \begin{bmatrix} 0 & \text{if} & k = 1, \\ \infty & \text{if} & k = 2, 3, \ldots, N; \end{bmatrix} \tag{4.19}$$

answers: $h_N(k)$ for $k = 1, 2, \ldots, N$. (4.20)

Problem 4.8. Show that $h_i(k)$, as defined by (4.17), is correctly given by (4.18) and (4.19).

Problem 4.9. Show that if $h_i(k) = h_{i-1}(k)$ $(k = 1, 2, \ldots, N)$ for some i, $2 \leqslant i \leqslant N$, then $h_i(k)$ is the length of the shortest path from node 1 to node k. Also show that if convergence does not occur by the Nth iteration, then a negative cycle exists. Therefore, procedure 3 never requires more than N iterations.

Problem 4.10. Find the shortest paths from node 1 to all nodes for the network given in Figure 4.4 using procedure 3.

Each iteration of procedure 3 requires $N(N - 1)/2$ additions and $N(N - 1)/2$ comparisons. On the other hand, each iteration of procedures 1 or 2 requires $N(N - 1) + 1$ additions and $N(N - 2) + 1$ comparisons. Therefore, each iteration of procedure 3 requires approximately half the number of operations required by each iteration of procedures 1 or 2.

Problem 4.11. Show that $h_{2i-1}(k) \leqslant g_i(k)$ $(k = 1, 2, \ldots, N)$ for $i = 1, 2, \ldots$. Also show that if procedure 2 converges in i iterations, then procedure 3 converges in no more than $2i - 2$ iterations. It follows from this latter result and the above statements that the number of operations required by procedure 3 is less than the number required by procedure 2. Furthermore, the worst case for procedure 3 requires approximately half the number of operations required by the worst case for procedure 2 (or 1). However, the worst cases will occur for different sets of data.

We know from Problems 4.7 and 4.11 that procedure 3 is the most efficient of the three procedures. However, in order to get a better idea of their relative efficiencies, we decided to apply the three procedures to some randomly generated networks. In particular, we generated 50 networks each with 20 nodes. The probability of an arc being positive was 0.99. Each positive arc was an integer-valued random variable uniformly distributed between 1 and 100. Each negative arc was an integer-valued random variable uniformly distributed between -1 and -10. For 32 out of the 50 networks, legitimate shortest paths were found. For the remaining 18 a negative cycle was found. In Table 4.1 we give the average number of operations required by the three procedures. In the case of no negative cycles, it can be seen that procedure 3 requires approximately 40% the number of operations required by procedure 1, and approximately 56% the number required by procedure 2.

Table 4.1. *Average number of operations required by the three procedures when applied to 50 randomly generated 20-node networks*

	Procedure 1	Procedure 2	Procedure 3
No negative cycles	4916	3501	1959
Negative cycles	14840	14840	7600

B. The Shortest Path between Each Pair of Nodes

We are given a network with N nodes, numbered arbitrarily from 1 to N. The problem is to find the shortest path from node j to node k for all j and k.

Once again, we shall initially assume that $d_{ij} \geqslant 0$ for every arc (i, j). One procedure for solving the stated problem is to apply Dijkstra's procedure with the list-processing technique suggested by Yen (hereafter called the modified Dijkstra procedure) N times, once for each possible source node. Since the modified Dijkstra procedure requires approximately $3N^2/2$ operations to solve the single-pair problem, it can solve the all-pairs problem in $3N^3/2$ operations.

An alternative procedure has been proposed by Spira [6]. It finds all shortest paths by using a variant of Dijkstra's procedure to find first the shortest paths from node 1 to all nodes, then find the shortest paths from node 2 to all nodes, etc. We shall not give a statement of Spira's procedure here because it involves issues that are not completely germane to dynamic programming. It should be noted that there are a number of minor errors in Spira's statement of the procedure which are corrected in Carson and Law [1].

Let us briefly discuss the efficiency of Spira's procedure as compared to the modified Dijkstra procedure. In [6], Spira has shown that if his procedure is properly implemented, then the number of operations it requires to solve the all-pairs problem is of the order $N^2 \log_2^2 N$. Thus, for large N Spira's procedure requires fewer operations than the modified Dijkstra procedure. Carson and Law [1] have empirically compared the two procedures using both computation time and number of operations as criteria for comparison. They found that for "large" networks (approximately 100 nodes or greater) Spira's procedure is better (requires less computation time), but that for "small" networks the modified Dijkstra procedure is superior. They also found that number of operations is a poor measure of their relative efficiency. For example, when $N = 120$

the modified Dijkstra procedure requires approximately 3 times as many operations, but only about 10% more computation time.

Let us now consider the all-pairs problem when one or more of the d_{ij} are negative. We shall first present two well-known procedures which have long been thought to have the same computational requirements. We shall then show that one of the procedures, when slightly modified, is actually considerably more efficient. All of the procedures that we consider can detect the existence of negative cycles.

The first procedure is due to Floyd [4]. It constructs optimal paths by inserting nodes, when appropriate, into more direct paths. Define the optimal value function $f_i(j, k)$ as follows:

$f_i(j, k)$ = the length of the shortest path from node j to node k when only paths with intermediate nodes belonging to the set of nodes $\{1, 2, \ldots, i\}$ are allowed ($i = 0$ corresponds to the directed arc from node j to node k). (4.21)

Then the dynamic-programming formulation is as follows:

recurrence relation:

$$f_i(j, k) = \min \begin{bmatrix} f_{i-1}(j, k) \\ f_{i-1}(j, i) + f_{i-1}(i, k) \end{bmatrix} \quad (i = 1, 2, \ldots, N), \quad (4.22)$$

boundary conditions: $f_0(j, k) = d_{jk},$ (4.23)

answers: $f_N(j, k)$ for $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N$. (4.24)

Notice that we have allowed the possibility of $j = k$ on the r.h.s. of (4.22). This allows us to detect negative cycles. If $f_i(j, j) < 0$ for some node, j, then a negative cycle exists.

Let us show that $f_i(j, k)$, as defined by (4.21), is correctly given by (4.22) and (4.23). We shall use mathematical induction, but once again we give only the inductive step. Assume that $f_{i-1}(j, k)$ is correctly given for all j and k by (4.22) and that the procedure did not stop at stage $i - 1$. We must show the same for $f_i(j, k)$. If the shortest path under consideration does not contain node i, then $f_i(j, k) = f_{i-1}(j, k)$. If node i is an intermediate node, then the shortest path (at stage i) from node j to node i uses only some subset of the nodes $\{1, 2, \ldots, i - 1\}$ as intermediate nodes (provided that $f_{i-1}(i, j) + f_{i-1}(j, i) \geq 0$ for every j, which is implied by $f_i(j, j) \geq 0$ for every j). A similar statement is true about the shortest path from node i to node k. Therefore, by our inductive hypothesis, $f_i(j, k) = f_{i-1}(j, i) + f_{i-1}(i, k)$.

If we define $p_i(j, k)$ to be the first intermediate node on the shortest path from node j to node k using $\{1, 2, \ldots, i\}$ as intermediate nodes, then

$p_i(j, k)$ can be computed at the same time as $f_i(j, k)$ from the following recursive relationship:

$$p_i(j, k) = \begin{bmatrix} p_{i-1}(j, k) & \text{if node } i \text{ is not on the shortest path} \\ & \text{using } \{1, 2, \ldots, i\} \text{ as intermediate nodes,} \\ \\ p_{i-1}(j, i) & \text{otherwise,} \end{bmatrix}$$

where $p_0(j, k) = k$.

Let us compute the number of operations required by Floyd's procedure. For each value of i ($i = 1, 2, \ldots, N$) we must evaluate $f_i(j, k)$ for $(N - 1)^2$ pairs of nodes j, k. (Note that if $j = i$ or $k = i$, then $f_i(j, k) = f_{i-1}(j, k)$.) Furthermore, for each evaluation of $f_i(j, k)$, one addition and one comparison are required. Therefore, a total of $N(N - 1)^2$ additions and $N(N - 1)^2$ comparisons, or $2N(N - 1)^2$ operations, are required.

Problem 4.12. Find all shortest paths using Floyd's procedure for the network given in Figure 4.5. All arcs without arrowheads are considered undirected, i.e., $d_{ij} = d_{ji}$.

The second procedure that we consider is due to Dantzig [2]. It differs from Floyd's procedure in that at iteration i, only paths (not necessarily optimal) between nodes numbered between 1 and i have been constructed. Define the optimal value function $g_i(j, k)$ (for $j = 1, 2, \ldots, i$ and $k = 1, 2, \ldots, i$) as follows:

$$g_i(j, k) = \text{the length of the shortest path from node } j \text{ to node}$$
$$k \text{ when only paths with intermediate nodes}$$
$$\text{belonging to the set of nodes } \{1, 2, \ldots, i\} \text{ are}$$
$$\text{allowed.} \qquad (4.25)$$

The dynamic-programming formulation of Dantzig's procedure is

recurrence relations:

$$g_i(j, i) = \min_{l = 1, \ldots, i-1} [g_{i-1}(j, l) + d_{li}] \qquad (j = 1, 2, \ldots, i - 1),$$

$$g_i(i, k) = \min_{l = 1, \ldots, i-1} [d_{il} + g_{i-1}(l, k)] \qquad (k = 1, 2, \ldots, i - 1),$$

$$g_i(i, i) = \min \begin{bmatrix} 0 \\ \min_{l = 1, \ldots, i-1} \{g_i(i, l) + g_i(l, i)\} \end{bmatrix}, \qquad (4.26)$$

$$g_i(j, k) = \min \begin{bmatrix} g_{i-1}(j, k) \\ g_i(j, i) + g_i(i, k) \end{bmatrix}$$

$$(j = 1, 2, \ldots, i - 1; \quad k = 1, 2, \ldots, i - 1);$$

-2

2   4

1   3   5

1   4   7   6

8   9

2   3

3   -5   5

6
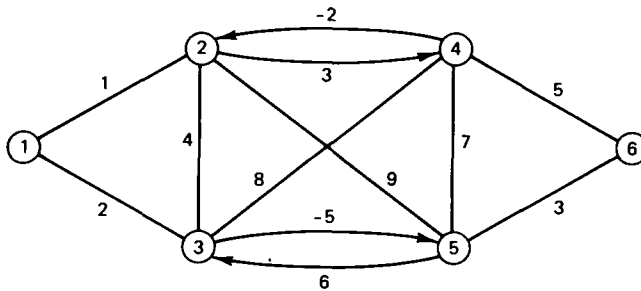
Figure 4.5

boundary condition: $g_1(1, 1) = 0;$ (4.27)

answers: $g_N(j, k)$ for $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N.$ (4.28)

The above recurrence relations are computed for $i = 2, 3, \ldots, N.$

Dantzig's procedure detects negative cycles by computing $g_i(i, i)$. If $g_i(i, i) < 0$, then a negative cycle exists. Furthermore, the correctness of Dantzig's procedure can be shown in a manner similar to the way we showed the correctness of Floyd's procedure.

Problem 4.13. Show that $N(N-1)(2N-1)/2$ additions and $N(N-1) \cdot (2N-3)/2$ comparisons, or $2N(N-1)^2$ operations, are required by Dantzig's procedure. (Note that if $j = k$, then $g_i(j, k) = g_{i-1}(j, k)$.)

Problem 4.14. Find all shortest paths using Dantzig's procedure for the network given in Figure 4.5.

We have seen above that Floyd's procedure and Dantzig's procedure both require approximately $2N^3$ operations. We now show that by slightly modifying Dantzig's procedure (see Tabourier [7]), it can be made considerably more efficient. The modified Dantzig procedure (which we state in the form of an algorithm) is as follows:

Step 0: $i = 2$, $g_1(1, 1) = 0.$

Step 1:   A. $l = 1$, $g_i^0(j, i) = \infty$ $(j = 1, 2, \ldots, i-1)$, $g_i^0(i, k) = \infty$ $(k = 1, 2, \ldots, i-1)$.

B.   If $d_{li} > g_i^{l-1}(l, i)$, then $g_i^l(j, i) = g_i^{l-1}(j, i)$ for $j = 1, 2, \ldots, i-1$ and go to step 1C. Otherwise, $g_i^l(j, i) = \min[g_i^{l-1}(j, i), g_{i-1}(j, l) + d_{li}]$ for $j = 1, 2, \ldots, i-1$.

C.   If $d_{il} > g_i^{l-1}(i, l)$, then $g_i^l(i, k) = g_i^{l-1}(i, k)$ for $k = 1, 2, \ldots, i-1$ and go to step 1D. Otherwise, $g_i^l(i, k) = \min[g_i^{l-1}(i, k), d_{il} + g_{i-1}(l, k)]$ for $k = 1, 2, \ldots, i-1$.

D. $l = l + 1$. If $l < i$, then go to step 1B. Otherwise, $g_i(j, i) =$

$g_i^{i-1}(j, i)$ $(j = 1, 2, \ldots, i - 1)$ and $g_i(i, k) = g_i^{i-1}(i, k)$ $(k = 1, 2, \ldots, i - 1)$. Go to step 2.

Step 2: If $g_i(i, l) + g_i(l, i) \geq 0$ for $l = 1, 2, \ldots, i - 1$, then $g_i(i, i) = 0$ and go to step 3. Otherwise, stop. A negative cycle exists.

Step 3: Compute $g_i(j, k) = \min[g_{i-1}(j, k), g_i(j, i) + g_i(i, k)]$ for $j = 1, 2, \ldots, i - 1$ and $k = 1, 2, \ldots, i - 1$.

Step 4: $i = i + 1$. If $i \leq N$, then go to step 1. Otherwise, stop.

We need to show that $g_i(j, i)$ is correctly computed from step 1. (The justification for $g_i(i, k)$ is similar.) It is easy to see that $g_i(j, i)$, as given by (4.26), can be computed as follows:

For $j = 1, 2, \ldots, i - 1$; let $g_i^0(j, i) = \infty$, and for $l = 1, 2, \ldots, i - 1$, compute

$$g_i^l(j, i) = \min\left[g_i^{l-1}(j, i), g_{i-1}(j, l) + d_{li}\right]$$

and set $g_i(j, i) = g_i^{i-1}(j, i)$.

However, it is clear that the loops over j and l can be interchanged and, thus, $g_i(j, i)$ for $j = 1, 2, \ldots, i - 1$ can also be computed as follows:

Let $g_i^0(j, i) = \infty$ for $j = 1, 2, \ldots, i - 1$. For $l = 1, 2, \ldots, i - 1$ and $j = 1, 2, \ldots, i - 1$, compute

$$g_i^l(j, i) = \min\left[g_i^{l-1}(j, i), g_{i-1}(j, l) + d_{li}\right].$$

Then set $g_i(j, i) = g_i^{i-1}(j, i)$ for $j = 1, 2, \ldots, i - 1$.

It remains to show that if $d_{li} \geq g_i^{l-1}(l, i)$, then $g_i^l(j, i) = g_i^{l-1}(j, i)$ for $j = 1, 2, \ldots, i - 1$. If d_{li} is infinite, then the result is clearly correct. If d_{li} is finite and $d_{li} \geq g_i^{l-1}(l, i)$, then $g_i^{l-1}(l, i)$ is finite and there exists an m, $1 \leq m < l$, such that $g_i^{l-1}(l, i) = g_{i-1}(l, m) + d_{mi} \leq d_{li}$. Then

$$g_i^{l-1}(j, i) \leq g_{i-1}(j, m) + d_{mi} \leq g_{i-1}(j, l) + g_{i-1}(l, m) + d_{mi}$$
$$\leq g_{i-1}(j, l) + d_{li}.$$

We have therefore shown the validity of the modified Dantzig procedure. In order to compare its computational efficiency with those of the previous two procedures, we randomly generated 50 20-node networks

Table 4.2. *Average number of operations required by the modified Dantzig procedure, by Dantzig's procedure, and by Floyd's procedure when applied to 50 randomly generated 20-node networks*

	Modified Dantzig procedure	Dantzig's procedure	Floyd's procedure
No negative cycles	9240	14440	14440
Negative cycles	4118	6449	6287

with the same characteristics as the networks considered in Section 3A. For 31 out of 50 networks, legitimate shortest paths were found. For the remaining 19, a negative cycle was detected. In Table 4.2 we give the average number of operations required by the modified Dantzig procedure, by Dantzig's procedure, and by Floyd's procedure. On the basis of the results given in Table 4.2, it would appear that the modified Dantzig procedure is considerably more efficient than the other two procedures.

Problem 4.15. Consider a general network with N nodes and d_{ij} that may be positive or negative. Define the value of a path to be the maximum d_{ij} along the path. Give an efficient procedure for finding the minimum value paths from node 1 to all other nodes.

Problem 4.16. Suppose you wish to route a message from node 1 to node N in a network, where p_{ij} is the known probability that a message routed from node i to node j arrives at node j. (Otherwise, the message is lost forever.) Give an efficient procedure for determing the path from node 1 to node N which has the maximal probability of arrival of the message at node N.

Problem 4.17. Consider a general network with N nodes. Let $d_{ij}(t)$ be the time required to travel from node i to node j when t is the time of departure from node i. Give an efficient procedure for determining the earliest possible arrival time at each node when you leave node 1 (the initial node) at time 0.

Problem 4.18. Modify Dijkstra's procedure to find the shortest path from node 1 to node 9 for the network given in Figure 4.6, where an additional cost of 2 is assessed for each turn. Hint: Associate with each node as many numbers as there are arcs leading into it. Even when one number at a node becomes permanent, the remaining temporary ones must continue to be processed as such.
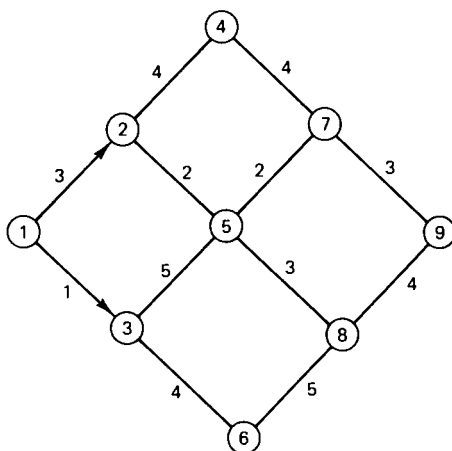
Figure 4.6

Problem 4.19.   Consider a general network with N nodes and nonnegative d_{ij}. Give an efficient Dijkstra-like procedure for finding the shortest path connecting some node in a specified subset A of the nodes to some node in a disjoint specified subset B of the nodes. (The initial and terminal nodes are to be chosen optimally.) How many additions and comparisons are required for the solution?

References

[1]   Carson, J. S., and A. M. Law, "A Note on Spira's Algorithm for the All-Pairs Shortest-Path Problem," Department of Industrial Engineering, Univ. of Wisconsin, Tech. Rep. 76-3 (1976).

[2]   Dantzig, G. B., "All Shortest Routes in a Graph," Operations Research House, Stanford Univ. Tech. Rep. 66-3 (1966).

[3]   Dijkstra, E. W., "A Note on Two Problems in Connexion with Graphs," *Numer. Math.* **1** (1959), 269–271.

[4]   Floyd, R. W., "Algorithm 97, Shortest Path," *Comm. ACM* **5** (1962), 345.

[5]   Ford, L. R., Jr., "Network Flow Theory," The Rand Corporation, P-293 (1956).

[6]   Spira, P. M., "A New Algorithm for Finding All Shortest Paths in a Graph of Positive Arcs in Average Time $O(n^2 \log^2 n)$," *SIAM J. Comput.* **2** (1973), 28–32.

[7]   Tabourier, Y., "All Shortest Distances in a Graph. An Improvement to Dantzig's Inductive Algorithm," *Discrete Math.* **4** (1973), 83–87.

[8]   Williams, T. A., and G. P. White, "A Note on Yen's Algorithm for Finding the Length of All Shortest Paths in N-Node Nonnegative-Distance Networks," *J. Assoc. Comput. Mach.* **20** (1973), 389–390.

[9]   Yen, J. Y., "An Algorithm for Finding Shortest Routes from All Source Nodes to a Given Destination in General Networks," *Quart. Appl Math.* **27** (1970), 526–530.

[10]   Yen, J. Y., "Finding the Lengths of All Shortest Paths in N-Node Nonnegative-Distance Complete Networks Using $\frac{1}{2} N^3$ Additions and N^3 Comparisons," *J. Assoc. Comput. Mach.* **19** (1972), 423–424.

Chapter 5

THE TRAVELING-SALESMAN PROBLEM

1. Introduction

In this chapter we consider the so-called traveling-salesman problem. We have a general network consisting of N cities (or nodes) such as the one described at the beginning of Chapter 4. The cities are arbitrarily numbered from 1 to N, and d_{ij} is the distance (or travel time, or cost, etc.) from city i to city j. A salesman would like to begin at city 1, visit each of the other cities once and only once, and then return to city 1 (such a path is called a *tour*). In what order should he visit the cities so that the total distance he travels is minimized?

This optimization problem and its variants are similar to problems actually faced by many companies, e.g., drawing up routes for delivery trucks. Furthermore, the same mathematical model occurs in other contexts such as production scheduling. Consider a manufacturer who would like to schedule optimally N different jobs on a piece of equipment. Let d_{ij} represent the set-up time required when job j is scheduled after job i. The d_{ij} may vary greatly and, in general, $d_{ij} \neq d_{ji}$. The manufacturer would like to schedule his production to minimize the total amount of set-up time.

The traveling-salesman problem can be considered a special case of the shortest-path problem studied in Chapter 4. However, since the traveling-salesman problem has received so much attention in the operations research literature, we decided to devote an entire chapter to its study.

2. Dynamic-Programming Formulation

In this section we present a dynamic-programming formulation for the traveling-salesman problem. Let $N_j = \{2, 3, \ldots, j - 1, j + 1, \ldots, N\}$ and

let S be a subset of N_j containing i members. Define the optimal value function $f_i(j, S)$ as

$f_i(j, S)$ = the length of the shortest path from city 1 to city j via the set of i intermediate cities S (the stage variable i indicates the number of cities in S).   (5.1)

The dynamic-programming formulation is

recurrence relation:

$$f_i(j, S) = \min_{k \in S} \left[f_{i-1}(k, S - \{k\}) + d_{kj} \right]$$
$$(i = 1, 2, \ldots, N - 2; \quad j \neq 1; \quad S \subseteq N_j);   (5.2)$$

boundary condition: $f_0(j, \_) = d_{1j};$ (5.3)

answer: $\min_{j = 2, 3, \ldots, N} \left[f_{N-2}(j, N_j) + d_{j1} \right].$ (5.4)

Let us intuitively justify the recurrence relation. Suppose that we would like to compute $f_i(j, S)$. Consider the shortest path from city 1 to city j via S which has city k (a member of S) immediately preceding city j. Since the cities in $S - \{k\}$ must be visited in an optimal order, the length of this path is $f_{i-1}(k, S - \{k\}) + d_{kj}$. Furthermore, since we are free to choose city k optimally, it is clear that $f_i(j, S)$ is correctly given by (5.2).

Therefore, to compute the length of the shortest tour, we begin by computing $f_1(j, S)$ (for every j, S pair) from the values of f_0. We then compute $f_2(j, S)$ (for every j, S pair) from the values of f_1. We continue in the same manner until $f_{N-2}(j, N_j)$ has been computed for every city j. The length of the shortest tour is then given by (5.4), since some city, j, must be the last city visited before returning to city 1. The corresponding shortest tour can be obtained by recording, for each stage and state, that city, k, which minimizes the r.h.s. of (5.2).

Let us consider an example. In Table 5.1 we give the distance matrix for a network consisting of five cities. The calculations for this problem are

Table 5.1.   *The distance from city i to city j, d_{ij}, for a network of five cities*

i \ j	1	2	3	4	5
1	0	3	1	5	4
2	1	0	5	4	3
3	5	4	0	2	1
4	3	1	3	0	3
5	5	2	4	1	0

as follows (the optimal policy function is given in parentheses):

$$f_0(2, \_) = d_{12} = 3(1),$$
$$f_0(3, \_) = 1(1),$$

$$f_0(4, \_) = 5(1),$$
$$f_0(5, \_) = 4(1).$$

$$f_1(2, \{3\}) = f_0(3, \_) + d_{32} = 1 + 4 = 5(3),$$
$$f_1(2, \{4\}) = 5 + 1 = 6(4),$$
$$f_1(2, \{5\}) = 4 + 2 = 6(5),$$
$$f_1(3, \{2\}) = 3 + 5 = 8(2),$$
$$f_1(3, \{4\}) = 5 + 3 = 8(4),$$
$$f_1(3, \{5\}) = 4 + 4 = 8(5),$$
$$f_1(4, \{2\}) = 3 + 4 = 7(2),$$
$$f_1(4, \{3\}) = 1 + 2 = 3(3),$$
$$f_1(4, \{5\}) = 4 + 1 = 5(5),$$
$$f_1(5, \{2\}) = 3 + 3 = 6(2),$$
$$f_1(5, \{3\}) = 1 + 1 = 2(3),$$
$$f_1(5, \{4\}) = 5 + 3 = 8(4).$$

$$f_2(2, \{3, 4\}) = \min\left[f_1(3, \{4\}) + d_{32}, f_1(4, \{3\}) + d_{42} \right]$$
$$= \min\left[8 + 4, 3 + 1 \right] = 4(4),$$
$$f_2(2, \{3, 5\}) = \min\left[8 + 4, 2 + 2 \right] = 4(5),$$
$$f_2(2, \{4, 5\}) = \min\left[5 + 1, 8 + 2 \right] = 6(4),$$
$$f_2(3, \{2, 4\}) = \min\left[6 + 5, 7 + 3 \right] = 10(4),$$
$$f_2(3, \{2, 5\}) = \min\left[6 + 5, 6 + 4 \right] = 10(5),$$
$$f_2(3, \{4, 5\}) = \min\left[5 + 3, 8 + 4 \right] = 8(4),$$
$$f_2(4, \{2, 3\}) = \min\left[5 + 4, 8 + 2 \right] = 9(2),$$
$$f_2(4, \{2, 5\}) = \min\left[6 + 4, 6 + 1 \right] = 7(5),$$
$$f_2(4, \{3, 5\}) = \min\left[8 + 2, 2 + 1 \right] = 3(5),$$
$$f_2(5, \{2, 3\}) = \min\left[5 + 3, 8 + 1 \right] = 8(2),$$
$$f_2(5, \{2, 4\}) = \min\left[6 + 3, 7 + 3 \right] = 9(2),$$
$$f_2(5, \{3, 4\}) = \min\left[8 + 1, 3 + 3 \right] = 6(4).$$

$$f_3(2, \{3, 4, 5\}) = \min\left[f_2(3, \{4, 5\}) + d_{32}, f_2(4, \{3, 5\}) \right.$$
$$\left. + d_{42}, f_2(5, \{3, 4\}) + d_{52} \right]$$
$$= \min\left[8 + 4, 3 + 1, 6 + 2 \right] = 4(4),$$
$$f_3(3, \{2, 4, 5\}) = \min\left[6 + 5, 7 + 3, 9 + 4 \right] = 10(4),$$
$$f_3(4, \{2, 3, 5\}) = \min\left[4 + 4, 10 + 2, 8 + 1 \right] = 8(2),$$
$$f_3(5, \{2, 3, 4\}) = \min\left[4 + 3, 10 + 1, 9 + 3 \right] = 7(2).$$

(equations continue)

$$\text{answer} = \min_{j=2,\,\ldots,\,5} \left[f_3(j, \{2, 3, 4, 5\} - \{j\}) + d_{j1} \right]$$
$$= \min[4 + 1, \, 10 + 5, \, 8 + 3, \, 7 + 5] = 5(2).$$

Therefore, the length of the shortest tour is 5 and the corresponding shortest tour is 1–3–5–4–2–1.

We shall now determine the computational requirements of the above dynamic-programming formulation. At stage i, $f_i(j, S)$ must be evaluated for $(N - 1)\binom{N-2}{i}$ different j, S pairs. Each such evaluation requires i additions and $i - 1$ comparisons. Therefore, for all stages the following number of additions and comparisons are required:

number of additions

$$= (N - 1) \sum_{i=1}^{N-2} i\binom{N-2}{i}$$

$$= (N - 1)(N - 2) \sum_{i=1}^{N-2} \frac{(N - 3)!}{(i - 1)!(N - 2 - i)!}$$

$$= (N - 1)(N - 2) \sum_{i=1}^{N-2} \binom{N-3}{i-1} = (N - 1)(N - 2) \sum_{j=0}^{N-3} \binom{N-3}{j}$$

$$= (N - 1)(N - 2)2^{N-3},$$

number of comparisons

$$= (N - 1) \sum_{i=1}^{N-2} (i - 1) \binom{N-2}{i}$$

$$= \text{number of additions} - (N - 1) \sum_{i=1}^{N-2} \binom{N-2}{i}$$

$$= (N - 1)(N - 2)2^{N-3} - (N - 1)(2^{N-2} - 1)$$

$$= (N - 1)2^{N-3}[(N - 2) - 2] + (N - 1) \approx (N - 1)(N - 4)2^{N-3}.$$

We have not considered the number of additions $(N - 1)$ and comparisons $(N - 2)$ necessary to compute the answer; however, these are negligible. Consider a 20-city traveling-salesman problem. For this problem a total of approximately 85 million operations are required.

Actually, the practical limit on the size of the problem that can be solved by the above procedure is probably due to storage space rather than the amount of computation. At stage i, $(N - 1)\binom{N-2}{i}$ storage locations are required to store f_i. For N even, the number of storage locations required will be maximum when $i = (N - 2)/2$. In the case of $N = 20$, approximately 925 thousand storage locations are necessary to store f_9.

Furthermore, to compute f_i, we must have all values of f_{i-1} available in core storage. Thus, it would seem that core storage large enough for any

two consecutive stages is necessary. However, this requirement can be avoided by the use of auxiliary storage. The values of f_i may be put in auxiliary storage as they are computed and later returned to core storage for calculation of f_{i+1} (f_{i-1} may be overwritten since it is no longer needed).

3. A Doubling-Up Procedure for the Case of Symmetric Distances

In this section we consider a doubling-up procedure for the traveling-salesman problem which is valid when the distance matrix is symmetric, i.e., when $d_{ij} = d_{ji}$ for all i, j. (The doubling-up procedure considered in this section is not exactly the same as the one presented in Chapter 1, Section 11.)

We shall assume that N is even (minor modifications are required when N is odd). Suppose that $f_1, f_2, \ldots, f_{(N-2)/2}$ have been computed from (5.2) and (5.3). Consider any tour that begins and ends at city 1. Some city, say j, will be visited at the halfway point. The length of the shortest path from city 1 to city j via any set of $(N-2)/2$ intermediate cities is given by $f_{(N-2)/2}$. Therefore, the length of the shortest tour is given by

$$\min_{j=2, 3, \ldots, N} \left[\min_{S \subseteq N_j} \{ f_{(N-2)/2}(j, S) + f_{(N-2)/2}(j, N_j - S) \} \right]. \quad (5.5)$$

Let us compare the computational requirements of the doubling-up procedure to those of the normal procedure. The following number of additions and comparisons are required by the doubling-up procedure:

$$\text{number of additions} = (N-1) \sum_{i=1}^{(N-2)/2} i \binom{N-2}{i}$$
$$+ (N-1) \frac{1}{2} \binom{N-2}{(N-2)/2},$$

$$\text{number of comparisons} = (N-1) \sum_{i=1}^{(N-2)/2} (i-1) \binom{N-2}{i}$$
$$+ (N-1) \frac{1}{2} \binom{N-2}{(N-2)/2} - 1,$$

where $\binom{N-2}{(N-2)/2}/2$ is the number of unordered partitions of a set of $N-2$ objects into two sets each of $(N-2)/2$ objects. In the case of $N = 20$, a total of approximately 43 million operations are required. Thus, the doubling-up procedure is considerably more efficient. On the other hand, the two procedures have exactly the same maximum storage requirements.

4.   Other Versions of the Traveling-Salesman Problem

We now consider two variants of the traveling-salesman problem which was considered in Section 2. Suppose we are interested in determining that path of minimum length which begins at city 1 and visits each of the other cities exactly once without returning to city 1. Then the answer to this problem is given by

$$\min_{j=2, 3, \ldots, N} \left[f_{N-2}(j, N_j) \right], \tag{5.6}$$

where $f_{N-2}(j, N_j)$ is as defined in Section 2.

Suppose now that we drop the requirement (in the problem of Section 2) that each city must be visited only once; i.e., we now desire that path of minimum length which begins at city 1, visits each of the other cities at least once, and then returns to city 1. In order to solve this problem, we replace each d_{ij} by d'_{ij}, where d'_{ij} is the length of the shortest path from city i to city j (the d'_{ij} may be computed by one of the shortest-path procedures presented in Chapter 4, Section 3B). We then solve the modified problem by the procedure of Section 2. If (i, j) is an arc in the optimal tour using the d'_{ij}, then (i, j) is replaced by the shortest path from city i to city j in order to find the optimal path for the problem we are now considering. Furthermore, for moderate to large values of N, the computational requirements for this procedure are approximately the same as those for the procedure of Section 2.

Problem 5.1.   Justify the procedure given in the preceding paragraph.

Problem 5.2.   Consider the traveling-salesman problem of Section 2. However, the cost of going from city i to city j depends, not only on i and j, but also on the total number of cities (excluding city 1) visited before city i. (Perhaps at each city a package is dropped off, making subsequent trips easier.) Let $d_{i, j, k}$ denote the cost of going from i to j when k cities have been visited previously. Give an efficient dynamic-programming formulation. How does the amount of computation compare to that required by the procedure of Section 2?

Problem 5.3.   Suppose in a traveling-salesman problem you start at city 1 and, except for the first trip (from 1 to some i) the cost of going from j to k depends also on the city visited immediately before j, call it p. That is, the given data are d_{1i} $(i = 2, 3, \ldots, N)$ and $d_{j, k, p}$ $(j = 2, 3, \ldots, N; k = 1, 2, \ldots, N; p = 1, 2, \ldots, N)$. Give a dynamic-programming formulation for finding the minimum-cost tour that visits each city once and only once.

Problem 5.4.   Consider a traveling-salesman problem where the path must visit each city once and only once, but need not return to the starting city. You are free to choose the starting city optimally. Give a more efficient dynamic-programming procedure than that of solving N problems, one for each possible starting city.

Problem 5.5. Suppose you are given N sets of cities, each set containing M cities. (The N sets are disjoint.) Let i_j denote city j ($j = 1, 2, \ldots, M$) in set i ($i = 1, 2, \ldots, N$) and d_{i_j, k_l} denote the distance from city j in set i to city l in set k. You wish to find the path of minimum length connecting a specified city A (not in any set of cities) to another specified city B (not in any set of cities) which visits one and only one city in each of the N sets of cities on the way.

Problem 5.6. Consider the traveling-salesman problem of Section 2 subject to the additional condition that precedence relations of the form "city i must precede city j in the path" must be satisfied for m (not necessarily a small number) specified pairs of cities i, j. Give an efficient dynamic-programming procedure. Is the amount of computation increased or decreased by the addition of the precedence relations? Explain. Use your *general procedure* to solve the problem given in Figure 5.1, where city 3 must precede city 2 (i.e., $m = 1$).

Figure 5.1

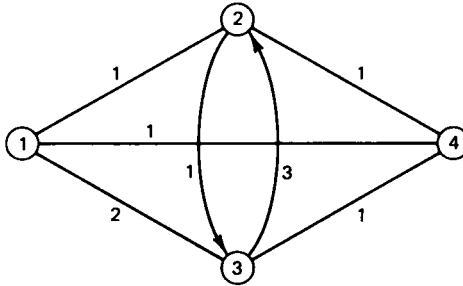**Problem 5.7.** Consider the traveling-salesman problem where you start at city 1 and then visit each of the other cities once and only once. (You do not return.) Give a Dijkstra-like procedure for solving this problem. Use this procedure to solve the problem given in Figure 5.2. Hint: 16 temporary labels, eight of which become permanent, must be calculated before the solution is known.
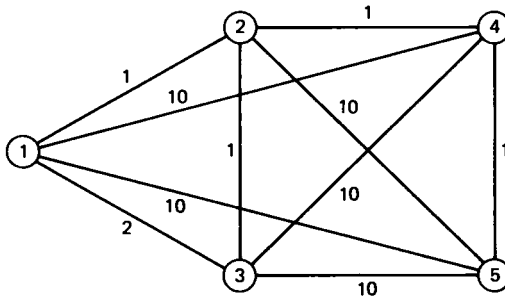
Figure 5.2

Chapter 6

PROBLEMS WITH LINEAR DYNAMICS
AND QUADRATIC CRITERIA

1. Introduction

Thus far we have considered specific problems and how to formulate them. Now we shall treat a quite general *class of problems* having a certain mathematical structure in common, a structure different from, but analogous to, the common structure of all linear-programming problems. For this class of problems, after the usual dynamic-programming formulation, we shall see that further mathematical analysis leads to a greatly simplified method of computational solution. This allows routine solution of problems with dozens of state variables, rather than the two or three state variables that can be handled in, say, resource allocation or equipment-replacement problems before numerical solution becomes prohibitively time consuming.

Few practical problems fall naturally into the class we are about to treat, but many can be so approximated with increasing accuracy at successive steps. This is analogous to the situation in linear programming, where most realistic problems do not exhibit the linear structure assumed, but sometimes nonlinear situations can be approximated fairly accurately by linear functions. To illustrate as simply as possible the kind of successive approximation scheme we have in mind, consider the calculus problem: what x minimizes $f(x)$ given by $f(x) = x^4 - 4x^3 + 8x^2 - 4x + 1$? Noting that setting the derivative of $f(x)$ equal to zero yields a cubic equation which is not easily solved, but also knowing that if $f(x)$ were quadratic, its derivative would be linear and when set equal to zero we *could* easily solve, we undertake to approximate $f(x)$ by a quadratic function. A few simple calculations tell us that the minimum of $f(x)$ occurs

76

somewhere near $x = 0$, so we take $x = 0$ as our first approximation x_0 of the minimizing x and construct that quadratic function that agrees with $f(x)$ in value and first two derivatives at $x = 0$. We do this by expanding $f(x)$ in a Taylor series about $x_0 = 0$, calling this approximating function $f_0(x)$, obtaining

$$f_0(x) = f(x_0) + f'(x)|_{x=x_0}(x - x_0) + \tfrac{1}{2} f''(x)|_{x=x_0}(x - x_0)^2 = 1 - 4x + 8x^2.$$

Now setting $f_0'(x)$ equal to zero we get

$$0 = -4 + 16x \qquad \text{or} \qquad x = \tfrac{1}{4}.$$

We take $\tfrac{1}{4}$ as our next approximation, called x_1, and repeat the above steps. The function $f_1(x)$ which agrees with $f(x)$ in value and first two derivatives at $x = \tfrac{1}{4}$ is

$$f_1(x) = \tfrac{113}{256} - \tfrac{11}{16}\left(x - \tfrac{1}{4}\right) + \tfrac{43}{8}\left(x - \tfrac{1}{4}\right)^2.$$

Hence, finding the value of x that minimizes $f_1(x)$, yields

$$0 = -\tfrac{11}{16} + \tfrac{43}{4}\left(x - \tfrac{1}{4}\right) \qquad \text{or} \qquad x = \tfrac{27}{86} = 0.31395.$$

While $x = 27/86$ is a fairly good approximation of the minimizing x, we could repeat the above procedure and further refine the answer, obtaining next $x = 0.31766$.

The successive approximation scheme that we have used here is called the Newton–Raphson procedure. It may not always converge and it may yield only a relative minimum, but for difficult problems beggars cannot be choosers. What is important is that it approximates, at each step, the actual problem by one that we can easily solve and, if all goes well, one which becomes a more accurate approximation of the actual problem in the neighborhood of its minimum at each step.

What we shall now develop is the dynamic-programming equivalent of the above procedure in which we solved a quadratic minimization problem by setting its derivative equal to zero and then solved a linear equation. We shall then ask the reader, in Problem 6.6, to use the procedure, as illustrated above, as the backbone of a successive approximation scheme.

2. A Linear Dynamics, Quadratic Criterion Model

We shall present and analyze a simple example of the type of problem that this chapter concerns, and then ask the reader, by means of problems, to treat more general versions.

Let $x(i)$ denote the state at stage i and let $y(i)$ denote the decision.

Then, we assume that the state at stage $i + 1$ is given by

$$x(i + 1) = g(i)x(i) + h(i)y(i), \qquad (6.1)$$

where $g(i)$ and $h(i)$, $i = 0, \ldots, N - 1$, are known constants. This is called a *linear dynamical system* since the rule giving the new state is linear in the old state and decision. Examples of linear dynamical systems abound. (In fact nonlinear dynamical systems are rare in operations research but common in such fields as aeronautics and chemical process control.) In resource allocation, where $x(i)$ is the total resource when we are considering activities i through N and $y(i)$ is the allocation to activity i, we have the linear dynamical equation

$$x(i + 1) = x(i) - y(i).$$

In inventory problems, if $x(i)$ is the inventory level starting stage i, $y(i)$ is the quantity ordered and immediately delivered, and $d(i)$ is the demand during stage i, we have the linear dynamical rule

$$x(i + 1) = x(i) + y(i) - d(i), \qquad \cdot$$

which is of form (6.1) except for the subtracted constant $d(i)$ and which can still be treated by the method to follow.

Unlike what has gone before, we assume that $x(i)$ and $y(i)$ are continuous variables that can assume any real values, either positive or negative. Hence we can use calculus.

The criterion is a summation of costs over N stages plus a terminal cost depending on $x(N)$, just as in most previous models. However, the cost of each stage is a quadratic function of $x(i)$ and $y(i)$, hence the designation *quadratic criterion*. We assume in this simple example that the criterion J is given by

$$J = \sum_{i=0}^{N-1} \left[a(i)x^2(i) + c(i)y^2(i) \right] + lx^2(N). \qquad (6.2)$$

In most operations research and other applications, the cost is *not* really quadratic. However, it can be approximated by a quadratic just as in the calculus example of Section 1 and then a method of successive approximation can be employed.

Given the state at stage zero, $x(0)$, our problem is to choose $y(0)$, $y(1), \ldots, y(N - 1)$ so as to minimize J given by (6.2) where the states $x(i)$ evolve by the rule (6.1).

3. A Particular Problem

To make precise the nature of our model, consider the problem: given $x(0) = 2$, choose $y(0)$, $y(1)$, and $y(2)$ so as to minimize J given by

$$J = y^2(0) + 12x^2(1) + 2y^2(1) + 2x^2(2) + y^2(2) + \tfrac{1}{4}x^2(3), \qquad (6.3)$$

where

$$x(1) = \tfrac{1}{2}x(0) + \tfrac{1}{6}y(0), \quad x(2) = 3x(1) + \tfrac{1}{2}y(1), \quad x(3) = 4x(2) + 2y(2).$$
(6.4)

This is a problem of the above type with: $a(0) = 0$, $c(0) = 1$, $a(1) = 12$, $c(1) = 2$, $a(2) = 2$, $c(2) = 1$, $l = \tfrac{1}{4}$, $g(0) = \tfrac{1}{2}$, $h(0) = \tfrac{1}{6}$, $g(1) = 3$, $h(1) = \tfrac{1}{2}$, $g(2) = 4$, $h(2) = 2$, and $N = 3$. A possible, but probably nonoptimal solution is $y(0) = 6$, $y(1) = 2$, $y(2) = \tfrac{1}{2}$. By (6.4) we obtain

$$x(1) = 1 + 1 = 2, \qquad x(2) = 6 + 1 = 7, \qquad x(3) = 28 + 1 = 29,$$

and hence

$$J = 36 + 48 + 8 + 98 + \tfrac{1}{4} + \tfrac{841}{4} = 400\tfrac{1}{2}.$$

Problem 6.1. A better solution is to make $x(1)$ equal to zero by choosing $y(0) = -6$ and then to make all subsequent y's equal to zero. Evaluate the cost of this solution.

We could solve the above problem by first solving for the x's in terms of the y's, obtaining,

$$x(1) = 1 + \tfrac{1}{6}y(0), \quad x(2) = 3 + \tfrac{1}{2}y(0) + \tfrac{1}{2}y(1),$$
$$x(3) = 12 + 2y(0) + 2y(1) + 2y(2).$$

We could then substitute for the x's in J, obtaining a quadratic function of the y's. Next we could set $\partial J/\partial y(0) = 0$, $\partial J/\partial y(1) = 0$, $\partial J/\partial y(2) = 0$ and solve the resulting three simultaneous linear equations for the minimizing y's.

Problem 6.2. Do so.

The procedure we shall now develop is easier and more systematic than the above, but will give the same result.

4. Dynamic-Programming Solution

We begin in the usual manner by defining the optimal value function $V_i(x)$ for the problem given by (6.1) and (6.2) by

$V_i(x) = $ the minimum cost of the remaining process if it starts stage i in state x.

Then, by the principle of optimality, for $i = 0, 1, \ldots, N - 1$,

$$V_i(x) = \min_y \left[a(i)x^2 + c(i)y^2 + V_{i+1}(g(i)x + h(i)y) \right]$$
(6.5)

with boundary condition

$$V_N(x) = lx^2. \tag{6.6}$$

We could (but will not) now use (6.6), given the data a, c, g, h, and l, to evaluate $V_N(x)$ at a discrete grid of points taken between some arbitrary upper and lower limits that we feel sure will include the optimal solution for the given initial condition. If $x(0) = 2$, we might use a grid consisting of $-5, -4.9, -4.8, \ldots, 0, 0.1, 0.2, \ldots, 4.9, 5$. Then we could determine $V_{N-1}(x)$ at these same points by either considering only those values of $y(N-1)$, given $x(N-1)$, that lead to the grid points at which $V_N(x)$ has been computed or else by using a fixed grid of reasonable decisions y (say $y = -5, -4.9, \ldots, 0, \ldots, 4.9, 5$) and when we need to know $V_N(x)$ at a point not actually computed, use an interpolation formula. That is, we could replace the problem by some sort of approximating discrete problem and proceed as in earlier chapters.

Instead we use calculus and actually calculate the exact expressions for $V_i(x)$ based on (6.5) and (6.6). By (6.5) and (6.6)

$$V_{N-1}(x) = \min_y \left[a(N-1)x^2 + c(N-1)y^2 + V_N(g(N-1)x + h(N-1)y) \right]$$

$$= \min_y \left[a(N-1)x^2 + c(N-1)y^2 + lg^2(N-1)x^2 \right.$$

$$\left. + 2lg(N-1)h(N-1)xy + lh^2(N-1)y^2 \right]. \tag{6.7}$$

Now we use calculus to determine the value of y, in terms of x, which minimizes the bracketed expression in (6.7). Setting the partial derivative with respect to y equal to zero,

$$0 = 2c(N-1)y + 2lg(N-1)h(N-1)x + 2lh^2(N-1)y,$$

$$y = -\frac{lg(N-1)h(N-1)x}{c(N-1) + lh^2(N-1)}. \tag{6.8}$$

This value of y yields the minimum if

$$c(N-1) + lh^2(N-1) > 0. \tag{6.9}$$

If the left-hand side of (6.9) equals zero and l, $g(N-1)$, $h(N-1)$, and x are nonzero, then $V_{N-1}(x)$ can be made $-\infty$ by suitably choosing y; and if it is negative, y given by (6.8) yields the maximum and no minimum exists (i.e., by choosing y as a sufficiently large positive or negative number the value of the bracketed expression in (6.7) can be made as small as desired). In these cases the problem is said to have no solution. Henceforth, we shall assume that (6.9) holds and that whenever we set a derivative equal to zero we can solve for y and that it is a minimizing y. There is no possibility of obtaining a relative but not absolute minimum since we are dealing with quadratic functions.

Knowing that the minimizing y is the linear function of the state x

given by (6.8), we substitute this y into (6.7) to obtain $V_{N-1}(x)$. This gives

$$V_{N-1}(x) = a(N-1)x^2 + c(N-1)\left[-\frac{lg(N-1)h(N-1)x}{c(N-1)+lh^2(N-1)}\right]^2$$

$$+ lg^2(N-1)x^2 + 2lg(N-1)h(N-1)x$$

$$\times \left[-\frac{lg(N-1)h(N-1)x}{c(N-1)+lh^2(N-1)}\right]$$

$$+ lh^2(N-1)\left[-\frac{lg(N-1)h(N-1)x}{c(N-1)+lh^2(N-1)}\right]^2$$

$$= \left[a(N-1)+lg^2(N-1)-\frac{l^2g^2(N-1)h^2(N-1)}{c(N-1)+lh^2(N-1)}\right]x^2$$

$$= p(N-1)x^2, \tag{6.10}$$

where we have denoted the coefficient of x^2, which can easily be computed from the given data, by $p(N-1)$. Note that the optimal value function for the process starting at stage $N-1$ is a quadratic function of the state x.

Now we wish to use (6.5), with $i = N-2$, to compute $V_{N-2}(x)$. If we write out the formula, using (6.10) for $V_{N-1}(x)$ on the right, we get exactly (6.7) except that l in (6.7) is replaced by $p(N-1)$. Hence we can write immediately that y at stage $N-2$ is given in terms of x at stage $N-2$ by (see (6.8))

$$y = -\frac{p(N-1)g(N-2)h(N-2)x}{c(N-2)+p(N-1)h^2(N-2)}, \tag{6.11}$$

and from (6.10) we deduce that $V_{N-2}(x)$ is given by

$$V_{N-2}(x) = p(N-2)x^2, \tag{6.12}$$

where

$$p(N-2) = a(N-2) + p(N-1)g^2(N-2)$$

$$-\frac{p^2(N-1)g^2(N-2)h^2(N-2)}{c(N-2)+p(N-1)h^2(N-2)}. \tag{6.13}$$

Clearly, we could repeat the process over and over again to get $V_{N-3}(x)$, $V_{N-4}(x)$, ..., $V_0(x)$. Our general result is that, for $i = 0, 1, \ldots, N-1$, $V_i(x)$ is given by

$$V_i(x) = p(i)x^2, \tag{6.14}$$

where $p(i)$ is determined recursively from $p(i+1)$ by

$$p(i) = a(i) + p(i+1)g^2(i) - \frac{p^2(i+1)g^2(i)h^2(i)}{c(i)+p(i+1)h^2(i)} \tag{6.15}$$

and the boundary condition

$$p(N) = l. \tag{6.16}$$

The optimal y at stage i, for given $x(i)$, is determined by

$$y(i) = -\frac{p(i+1)g(i)h(i)x(i)}{c(i) + p(i+1)h^2(i)}. \tag{6.17}$$

The elegant way of *proving* this result is by induction. Noting that $V_N(x)$ is lx^2, we assume that $V_{i+1}(x)$ has the form

$$V_{i+1}(x) = p(i+1)x^2 \tag{6.18}$$

and then, using (6.5) as we did in obtaining (6.7), (6.8), and (6.10), we show that $V_i(x)$ has the form (6.14) with (6.15) and (6.16) holding. To be sure that (6.17) gives the *minimizing* y we assume throughout, based on (6.9), that

$$c(i) + p(i+1)h^2(i) > 0. \tag{6.19}$$

Problem 6.3. Use the above formulas to solve the problem solved in Problem 6.2 and verify that the results agree on the optimal sequence $y(i)$.

Problem 6.4. Consider the minimization problem where the criterion J is the most general quadratic function

$$J = \sum_{i=0}^{N-1} \left[a(i)x^2(i) + b(i)x(i)y(i) + c(i)y^2(i) + d(i)x(i) + e(i)y(i) + f(i) \right]$$
$$+ lx^2(N) + wx(N) + z,$$

and the dynamics is the most general linear function

$$x(i+1) = g(i)x(i) + h(i)y(i) + k(i).$$

Prove by induction that the optimal value function $V_i(x)$ has the form

$$V_i(x) = p(i)x^2 + q(i)x + r(i) \qquad (i = 0, \ldots, N),$$

and determine recursive formulas for $p(i)$, $q(i)$, and $r(i)$.

Problem 6.5. Consider the following minimization problem in vector–matrix notation:

$$J = \sum_{i=0}^{N-1} \left[x^T(i)A(i)x(i) + x^T(i)B(i)y(i) + y^T(i)C(i)y(i) \right.$$
$$+ d(i)x(i) + e(i)y(i) + f(i) \Big]$$
$$+ x^T(N)Lx(N) + wx(N) + z,$$
$$x(i+1) = G(i)x(i) + H(i)y(i) + k(i) \qquad (i = 0, \ldots, N-1),$$

where $x(i)$ is an $n \times 1$ vector, $y(i)$ is an $m \times 1$ vector, $A(i)$ is a symmetric $n \times n$ matrix, $C(i)$ is a symmetric $m \times m$ matrix, $B(i)$ is $n \times m$, $d(i)$ is $1 \times n$, $e(i)$ is $1 \times m$, f is a scalar, L is a symmetric $n \times n$ matrix, w is $1 \times n$, z is a scalar, $G(i)$ is an $n \times n$ matrix, $H(i)$ is an $n \times m$ matrix, $k(i)$ is $n \times 1$, and T denotes the

transpose. The state dimension n is assumed greater than or equal to the decision dimension m. A, C, and L are assumed symmetric because a quadratic form $x^T Q x$ for any Q can always be rewritten as $\frac{1}{2} x^T (Q + Q^T) x$ and $(Q + Q^T)$ is symmetric.

Use matrix calculus to show that the optimal value function of n variables $V_i(x(i))$ has the form

$$V_i(x(i)) = x^T(i) P(i) x(i) + q(i) x(i) + r(i),$$

where $P(i)$ is a symmetric $n \times n$ matrix, $q(i)$ is $1 \times n$, and $r(i)$ is a scalar, and determine recursive formulas for $P(i)$, $q(i)$, and $r(i)$.

The computational convenience of the above dynamic-programming method of solution of this model is immense. For the completely general Problem 6.5 we must compute $\frac{1}{2} n(n+1) + n + 1$ numbers at each stage. To do so we must invert an $m \times m$ matrix (which requires roughly m^3 multiplications and m^3 additions) and we must multiply matrices of dimensions up to $n \times n$ (requiring n^3 multiplications and n^3 additions) about 10 times. Letting $m = n$, we get roughly $10n^3$ multiplications and $10n^3$ additions per stage. Let us take $5 \cdot 10^{-5}$ seconds as the time for one operation because multiplication is generally slower than addition and because doubly subscripted variables require considerable bookkeeping. Then, if $n = m = 20$, each stage would take roughly 7 seconds, not bad for a problem with 20 state variables and 20 decisions in each state at each stage.

Problem 6.6. Consider the nonquadratic criterion J given by

$$J = y^4(0) + x^4(1) + y^4(1) + x^4(2)$$

and nonlinear dynamics

$$x(i+1) = x^2(i) + 4y(i) \qquad (i = 0, 1),$$
$$x(0) = 2.$$

Start with the decision sequence $y(0) = -\frac{7}{8}$, $y(1) = -\frac{1}{16}$. Determine the x sequence given by the dynamics. Compute the value of J for these sequences. Calling the starting y sequence $\bar{y}$ and the resulting x sequence $\bar{x}$, approximate the cost at each stage by the Taylor series expansion about $(\bar{x}, \bar{y})$ through the quadratic terms,

$$f(x, y) \approx f(\bar{x}, \bar{y}) + \frac{\partial f}{\partial x}\bigg|_{\bar{x}, \bar{y}} (x - \bar{x}) + \frac{\partial f}{\partial y}\bigg|_{\bar{x}, \bar{y}} (y - \bar{y})$$

$$+ \frac{1}{2}\left[\frac{\partial^2 f}{\partial x^2}\bigg|_{\bar{x}, \bar{y}} (x - \bar{x})^2 + 2 \frac{\partial^2 f}{\partial x \, \partial y}\bigg|_{\bar{x}, \bar{y}} (x - \bar{x})(y - \bar{y}) \right.$$

$$\left. + \frac{\partial^2 f}{\partial y^2}\bigg|_{\bar{x}, \bar{y}} (y - \bar{y})^2 \right].$$

Approximate the right-hand side of the dynamics by the above series expansion about $(\bar{x}, \bar{y})$ through linear terms only. Use the method and formulas of Problem

6.4 to find the optimal decision rule for the resulting problem. Obtain new y and x sequences as follows. Determine $y(0)$ by the optimal decision rule. Obtain $x(1)$ by the actual nonlinear dynamics. Obtain $y(1)$ from the optimal decision rule using the actual $x(1)$. Obtain $x(2)$ from the nonlinear dynamics. Evaluate J for this new x, y sequence and if it is an improvement on the original J, repeat the entire process once more.

Problem 6.7. For a problem of the type of Problem 6.4, $r(i)$ will be zero for all i if and only if, when the initial state is zero, the optimal cost of the remaining process is zero. Why? $q(i)$ will be zero for all i if and only if the optimal cost is such that $V_i(x) = V_i(-x)$ for all x. Why?

By inspection of (6.1) and (6.2), deduce that $q(i)$ and $r(i)$ are zero for all i, as shown by (6.14).

Problem 6.8. In Problem 6.4, suppose that $x(0)$ is not specified and it, as well as the decision sequence, are to be chosen so as to minimize the criterion. How would you use the solution of Problem 6.4 to solve this problem?

Problem 6.9. Consider the problem: minimize J given by

$$J = lx^2(0) + wx(0) + z + \sum_{i=1}^{N} \left[a(i)x^2(i) + c(i-1)y^2(i-1) \right],$$

where

$$x(i+1) = g(i)x(i) + h(i)y(i) + k(i) \qquad (i = 0, \ldots, N-1)$$

and

$$x(0) \text{ is unspecified}, \qquad x(N) \text{ is given.}$$

Develop formulas for a forward dynamic-programming solution procedure.

Problem 6.10. Consider the problem: minimize J given by

$$J = x^2(0) + y^2(0) + x^2(1) + y^2(1) + x^2(2),$$

where

$$x(i+1) = x(i) + y(i) + 1 \qquad (i = 0, 1),$$

and neither $x(0)$ nor $x(2)$ is specified. Solve backward using the formulas of Problem 6.4 and the procedure of Problem 6.8. Solve forward using the formulas of Problem 6.9 and the idea of Problem 6.8. Verify that your answers agree.

Problem 6.11. Consider the dynamical system, without any decision variables,

$$x(i+1) = b(i)x(i) + d(i) \qquad (i = 0, 1, \ldots, N-1),$$

where the b's and d's are given. Suppose that $x(0)$ is unknown. (If it were known, the entire sequence of x's would be known.) You are given a sequence of "observed states" $z(0), \ldots, z(N)$ which are error-corrupted readings of $x(0), \ldots, x(N)$. You are to determine that value of $x(0)$ such that the resulting sequence $x(0), \ldots,$ $x(N)$, given by the dynamics, deviates as little as possible from the observed states

$z(0), \ldots, z(N)$ in the sense that the deviation cost J given by

$$J = \sum_{i=0}^{N} (z(i) - x(i))^2$$

is minimized. Give a forward dynamic-programming solution procedure, based on computing $F_k(x)$, $k = 0, \ldots, N$, where $F_k(x)$ is the deviation cost from stage 0 through stage k given that $x(k) = x$. Computation of $F_k(x)$ does not involve any minimization. The solution sequence is the one leading to the value of x that minimizes $F_N(x)$.

Problem 6.12. Develop a dynamic-programming procedure, similar to that of the text, for the time-lagged dynamics problem where (6.1) is replaced by

$$x(i + 1) = g_1(i)x(i) + g_2(i)x(i - 1) + h(i)y(i) \qquad (i = 1, \ldots, N - 1)$$

(with $x(0) = c_0$ and $x(1) = c_1$ specified) and the criterion is given by (6.2).

Problem 6.13. Develop a dynamic-programming procedure, similar to that of the text, for a problem with dynamics given by (6.1) but where the criterion depends not only on $y(i)$ but also on $y(i) - y(i - 1)$, the change in $y(i)$ from the previous value. Specifically

$$J = \sum_{i=1}^{N-1} \left[a(i)x^2(i) + c(i)y^2(i) + d(i)(y(i) - y(i - 1))^2 \right] + lx^2(N),$$

where $x(0)$ and $y(0)$ are specified.

Problem 6.14. Consider the following quadratic dynamics, linear criterion problem (note the reversal of the usual assumption):

minimize: $\sum_{i=0}^{N-1} [a(i)x(i) + c(i)y(i)] + lx(N)$

subject to $x(i + 1) = g(i)x^2(i) + h(i)y^2(i) \qquad (i = 0, \ldots, N - 1)$,
             $x(0)$ specified.

Is the optimal value function linear? quadratic? neither?

Problem 6.15. Consider the problem given by (6.1) and (6.2) with the additional constraint

$$-1 \leqslant y(i) \leqslant 1 \qquad (i = 0, \ldots, N - 1).$$

Try to develop a procedure like that of the text.

5. Specified Terminal Conditions

In the problem defined in Section 2 by (6.1) and (6.2), $x(N)$, the terminal value of x, was not specified, but a cost $lx^2(N)$ was attached to the terminal point. Let us see how the analysis is modified if we require that the decisions $y(i)$ be chosen so that $x(N) = t$, a given number. No cost

is associated with the terminal point since all admissible decision policies must lead to it. We first note that $V_N(x)$ is now defined only for $x = t$ since if we start stage N at any other point, there is no way of moving to the specified terminal point at the same stage. However $V_{N-1}(x)$ is defined for all x since by (6.1) we can determine, given $x(N - 1)$, the $y(N - 1)$ that yields $x(N) = t$. In terms of $x(N - 1)$ we have

$$t = g(N - 1)x(N - 1) + h(N - 1)y(N - 1) \qquad (6.20)$$

so (assuming $h(N - 1)$ is nonzero)

$$y(N - 1) = \frac{t - g(N - 1)x(N - 1)}{h(N - 1)} . \qquad (6.21)$$

The cost of stage $N - 1$, $V_{N-1}(x)$, if we start in state x and use (6.21) to determine y is

$$V_{N-1}(x) = a(N - 1)x^2 + c(N - 1)\left[\frac{t - g(N - 1)x}{h(N - 1)} \right]^2$$

$$= \left[a(N - 1) + \frac{c(N - 1)g^2(N - 1)}{h^2(N - 1)} \right]x^2$$

$$+ \left[-\frac{2c(N - 1)tg(N - 1)}{h^2(N - 1)} \right]x + \left[\frac{c(N - 1)t^2}{h^2(N - 1)} \right]. \quad (6.22)$$

Hence $V_{N-1}(x)$ has the same form as in the solution of Problem 6.4 with $p(N - 1)$, $q(N - 1)$, and $r(N - 1)$ given by the three bracketed expressions on the right in (6.22). Now the recursive formulas derived in the solution of Problem 6.4, with $b = d = e = f = k = 0$, can be used to solve the problem.

The point here is that for this new problem the optimal value function is still quadratic and the optimal decision linear in the state x, only the terminal conditions for the recurrence relations for the coefficients of the optimal value function are affected.

Problem 6.16. Do the problem of Section 3 ignoring the terminal cost term $\frac{1}{4}x^2(3)$, with the terminal condition $x(3) = 2$.

The above procedure works well when the dimension of the state vector (1 in the above problem) and the dimension of the decision vector (also 1 in the above problem) are the same. Then, given certain assumptions about $h(N - 1)$, $y(N - 1)$ is uniquely determined by $x(N - 1)$ and the terminal condition. If the dimension of the state is an integer multiple z of the dimension of the decision (if we have four state variables and two decision variables, z would be 2), and if $h(i)$ has certain reasonable properties, the decisions y are uniquely determined during the

last z stages so $V_{N-z}(x)$ can be determined and can be used to start the backward solution.

Problem 6.17. Consider the problem: choose $y(0)$, $y(1)$, and $y(2)$ that minimize J given by

$$J = y^2(0) + x_1^2(1) + x_2^2(1) + y^2(1) + x_1^2(2) + x_2^2(2) + y^2(2)$$

subject to $\quad x_1(0) = 7, \quad x_2(0) = 1;$

$$x_1(3) = 2, \quad x_2(3) = 1;$$

where

$$x_1(i+1) = x_1(i) + x_2(i) + 2y(i), \qquad x_2(i+1) = x_1(i) - x_2(i) + y(i).$$

(Note that z as defined above is 2 for this example.) Determine $V_1(x_1, x_2)$ by using the fact that the terminal conditions determine $y(1)$ and $y(2)$ in terms of $x_1(1)$ and $x_2(1)$. Now use the formulas from Problem 6.5 to find $y(0)$ and $V_0(7, 1)$. Verify that the cost of the solution determined by your $y(0)$, $y(1)$, and $y(2)$ is indeed $V_0(7, 1)$.

The above procedure encounters difficulties for systems where the dimension n of x is not a multiple of the dimension m of y. For example, if $n = 3$, $m = 2$, we cannot get from an arbitrary x vector at stage $N - 1$ to a fixed point, since analogous to (6.20) we would have three equations in two unknowns. But for an arbitrary x vector starting stage $N - 2$ we have three equations in four unknowns (two components of y at each of two stages), which means $V_{N-2}(x_1, x_2, x_3)$ can be determined only by solving a constrained optimization problem. We now give a general procedure that works for any dimension of state and decision vectors, and for terminal conditions specifying linear relations among the state variables as well as for specified values of some or all of the components of the terminal state. We introduce the procedure by reconsidering the problem presented at the beginning of this section and by showing a quite different way of obtaining the result already deduced. Then, in problems, we ask the reader to extend the procedure to the more general situations we mentioned above.

First, we neglect the terminal condition $x(N) = t$, but instead add to the criterion the term $\lambda x(N)$, where λ is an as yet undetermined constant. We have thereby attached a cost λ to the terminal point, and we shall later determine λ so that the terminal point will be t, as required. This should remind the reader of the Lagrange multiplier procedure we introduced in Chapter 3, Section 6. However, now we shall determine the proper value of λ analytically rather than computationally.

The minimizing solution of the original, terminally constrained, problem must render this modified criterion stationary (i.e., the first partial derivatives with respect to the decision variables must equal zero), but as shown by example in Problem 6.22 it does not necessarily *minimize* the

modified criterion. However, since a multidimensional quadratic function takes on a unique value at all argument values making the first partial derivatives zero (there can be more than one such point), we can still proceed much as before.

Our modified problem is now, choose $y(0), \ldots, y(N - 1)$ that render stationary J given by

$$J = \sum_{i=0}^{N-1} \left[a(i)x^2(i) + c(i)y^2(i) \right] + \lambda x(N), \tag{6.23}$$

where

$$x(i + 1) = g(i)x(i) + h(i)y(i) \qquad (i = 0, \ldots, N - 1). \tag{6.24}$$

Recognizing that the value of the solution to this problem depends on the initial stage, initial state, *and our choice of* λ, we define the value function, for $i = 0, \ldots, N$, by

$W_i(x, \lambda) =$ the stationary value of the remaining cost if we start stage i in state x with terminal cost $\lambda x(N)$.

We have the recurrence relation

$$W_i(x, \lambda) = \operatorname*{stat}_y \left[a(i)x^2 + c(i)y^2 + W_{i+1}(g(i)x + h(i)y, \lambda) \right]$$
$$(i = 0, \ldots, N - 1) \tag{6.25}$$

(where *stat* means y such that the partial derivative with respect to y equals 0) with boundary condition

$$W_N(x, \lambda) = \lambda x. \tag{6.26}$$

We now assume, as an inductive hypothesis, that $W_{i+1}(x, \lambda)$ is given by the quadratic function of x *and* λ,

$$W_{i+1}(x, \lambda) = p(i + 1)x^2 + q(i + 1)\lambda x + r(i + 1)\lambda^2. \tag{6.27}$$

Clearly $W_N(x, \lambda)$ has this form, with

$$p(N) = 0, \qquad q(N) = 1, \qquad r(N) = 0 \tag{6.28}$$

and we shall show that if $W_{i+1}(x, \lambda)$ has this form, so does $W_i(x, \lambda)$.

Substituting (6.27) on the right in (6.25) we obtain

$$W_i(x, \lambda) = \operatorname*{stat}_y \left[a(i)x^2 + c(i)y^2 + p(i + 1)\left(g(i)x + h(i)y \right)^2 \right.$$
$$\left. + q(i + 1)\lambda\left(g(i)x + h(i)y \right) + r(i + 1)\lambda^2 \right]. \tag{6.29}$$

The y that renders stationary the bracketed expression in (6.29) is given by

$$y = - \frac{2p(i + 1)g(i)h(i)x + q(i + 1)h(i)\lambda}{2c(i) + 2p(i + 1)h^2(i)}. \tag{6.30}$$

Substituting this y into (6.29) and regrouping gives

$$W_i(x, \lambda) = \left[a(i) + g^2(i)p(i + 1) - \frac{g^2(i)h^2(i)p^2(i + 1)}{c(i) + p(i + 1)h^2(i)} \right] x^2$$

$$+ \left[g(i)q(i + 1) - \frac{g(i)h^2(i)p(i + 1)q(i + 1)}{c(i) + p(i + 1)h^2(i)} \right] \lambda x$$

$$+ \left[r(i + 1) - \frac{h^2(i)q^2(i + 1)}{4\big(c(i) + p(i + 1)h^2(i)\big)} \right] \lambda^2$$

$$= p(i)x^2 + q(i)\lambda x + r(i)\lambda^2. \qquad (6.31)$$

Hence we have proved our inductive hypothesis and can compute $p(i)$, $q(i)$, and $r(i)$, once we know $p(i + 1)$, $q(i + 1)$, and $r(i + 1)$, by (6.28) and (6.31). Note that p, q, and r do not depend on λ. However y given by (6.30) does depend on λ so the value of $x(N)$, given an initial i and x, is a function of λ.

To find how $x(N)$ depends on λ we use a procedure analogous to the one used to find $W_i(x, \lambda)$ except that, y being determined by (6.30), we have, as in Problem 6.11, a dynamic process *but not a decision problem*. We begin by defining

$T_i(x, \lambda) = $ the terminal value of x, $x(N)$, if we start stage i in
                state x with a given value of λ and use (6.30).

Since the terminal x, given initial stage i, state x, and λ, is the same as the terminal x starting stage $i + 1$ in the state $x(i + 1)$ that results if we start stage i in state x and use $y(i)$ given by (6.30), we can write

$$T_i(x, \lambda) = T_{i+1}(g(i)x + h(i)y(i), \lambda)$$

$$= T_{i+1}\left[g(i)x - h(i)\left(\frac{2p(i + 1)g(i)h(i)x + q(i + 1)h(i)\lambda}{2c(i) + 2p(i + 1)h^2(i)} \right), \lambda \right]$$

$$(i = 0, \ldots, N - 1). \qquad (6.32)$$

Clearly, by the definition of $T_i(x, \lambda)$, we have the boundary condition

$$T_N(x, \lambda) = x. \qquad (6.33)$$

We now make the inductive assumption that $T_{i+1}(x, \lambda)$ is given by the linear function

$$T_{i+1}(x, \lambda) = b(i + 1)x + d(i + 1)\lambda. \qquad (6.34)$$

Such is clearly the case for $T_N(x, \lambda)$ given by (6.33), with

$$b(N) = 1, \qquad d(N) = 0. \qquad (6.35)$$

Substituting (6.34) on the right in (6.32) we obtain

$$T_i(x, \lambda) = b(i + 1)\left[g(i)x - h(i)\left(\frac{2p(i + 1)g(i)h(i)x + q(i + 1)h(i)\lambda}{2c(i) + 2p(i + 1)h^2(i)} \right) \right]$$
$$+ d(i + 1)\lambda. \tag{6.36}$$

Regrouping,

$$T_i(x, \lambda) = \left[g(i)b(i + 1) - \frac{g(i)h^2(i)p(i + 1)b(i + 1)}{c(i) + p(i + 1)h^2(i)} \right]x$$

$$+ \left[d(i + 1) - \frac{q(i + 1)h^2(i)b(i + 1)}{2\big(c(i) + p(i + 1)h^2(i)\big)} \right]\lambda$$

$$= b(i)x + d(i)\lambda. \tag{6.37}$$

Consequently, we have proved our hypothesis and we have recurrence relations for $b(i)$ and $d(i)$.

Examining the recurrence relation and boundary condition for $q(i)$ given by (6.28) and (6.31), we see that $b(i)$ above satisfies the same recurrence relation and boundary condition. Hence

$$b(i) = q(i), \tag{6.38}$$

and we need not solve an additional recurrence relation, once we have obtained $p(i)$, $q(i)$, and $r(i)$, in order to find $b(i)$. Now substitute q for b in the recurrence relation given by (6.37) for $d(i)$. Since both $d(N)$ and $r(N) = 0$, and since there is a 4 in the denominator of the recurrence relation for r where there is a 2 for d but otherwise the formulas are the same, we conclude that

$$d(i) = 2r(i) \tag{6.39}$$

so, again, we know d once we have computed p, q, and r.

Our last step is to use (6.37) (with substitutions using (6.38) and (6.39)) to determine what λ, given the initial stage i and state x, yields $x(N) = t$. Since by definition, $T_i(x, \lambda)$ is the value of $x(N)$, we let $T_i(x, \lambda) = t$ and solve for λ, obtaining $t = q(i)x + 2r(i)\lambda$, hence

$$\lambda = \frac{t - q(i)x}{2r(i)} . \tag{6.40}$$

Given any initial state and stage we can determine λ by (6.40) and then repeatedly use (6.30) with this constant value of λ to find the sequence of y's.

The solution given by the above formulas will yield the absolute minimum unless the criterion can be made infinitely negative.

For the case of multidimensional state x and decision y, with y of

lower dimension than x, which we shall treat in subsequent problems, $r(i)$ is a matrix and r^{-1} will not exist until we are a sufficient number of stages back from the end such that it is possible to get from any initial state to any specified terminal state.

Problem 6.18. Verify that our general procedure is in agreement with results (6.21) and (6.22) at the start of this section as follows. Use (6.31) and 6.28) to compute $p(N-1)$, $q(N-1)$, and $r(N-1)$. Use (6.40) to obtain λ in terms of $x(N-1)$ and then (6.30) to get $y(N-1)$ in terms of $x(N-1)$. Use λ in terms of $x(N-1)$ to compute $W_{N-1}(x, \lambda)$ by (6.31) as a quadratic function of $x(N-1)$. Subtract λt from W to obtain (see (6.23)) $V_{N-1}(x)$.

Problem 6.19. Consider the multidimensional problem, in matrix notation,

$$J = \sum_{i=0}^{N-1} \left[x^{\mathrm{T}}(i)A(i)x(i) + y^{\mathrm{T}}(i)C(i)y(i) \right]$$

$$x(i+1) = G(i)x(i) + H(i)y(i) \qquad (i = 0, \ldots, N-1)$$

$$x(N) = z,$$

where the matrix A is symmetric and $n \times n$, C is symmetric and $m \times m$, G is $n \times n$, H is $n \times m$, and x is $n \times 1$, y is $m \times 1$, and the specified vector terminal condition z is $n \times 1$.

Let $W_i(x, \lambda)$ be the stationary remaining cost (including $x^{\mathrm{T}}(N)\lambda$) for the free terminal state problem where x and λ are $n \times 1$ vectors. Assume that

$$W_{i+1}(x, \lambda) = x^{\mathrm{T}}P(i+1)x + x^{\mathrm{T}}Q(i+1)\lambda + \lambda^{\mathrm{T}}R(i+1)\lambda,$$

where P and R are symmetric and P, Q, and R are $n \times n$ matrices.

Derive recurrence relations for $P(i)$, $Q(i)$, and $R(i)$, and the formula for $y(i)$. Determine the formula for λ in terms of $x(i)$ so that $x(N) = z$.

Problem 6.20. How would the result of Problem 6.19 be modified if, instead of a fixed terminal vector $x(N)$, the terminal vector were required to satisfy e linear equations ($e \leqslant n$)

$$Ex(N) = z,$$

where the given matrix E is $e \times n$ and the given vector z is $e \times 1$?

Problem 6.21. Use the formulas from Problem 6.19 to obtain the equation of $V_1(x_1, x_2)$ that you got in Problem 6.17.

Problem 6.22. Consider the problem: choose $y(0)$ and $y(1)$ that minimize J given by

$$J = y^2(0) + x^2(1) - \tfrac{1}{2} y^2(1)$$

subject to

$$x(i+1) = x(i) + y(i) \qquad (i = 0, 1),$$

$$x(0) = 1,$$

$$x(2) = 0.$$

Solve by the method of (6.20)–(6.22). Now modify the criterion by adding $\lambda x(2)$

and solve by the general method of (6.23)–(6.40). Note that $y(1)$ maximizes, rather than minimizes, the remaining modified cost for any fixed λ.

6. A More General Optimal Value Function

The usual backward method of dynamic-programming formulation solves problems for all possible initial stages and states, with the terminal conditions for the problems being those specified in the particular problem to be solved. The terminal conditions are reflected in the boundary conditions. Of course, it is also common to use the forward procedure, with initial conditions of the given problem determining the boundary conditions.

Sometimes it is desirable to solve problems for all possible values of both initial and terminal states. It turns out that this can be done for the model of this chapter at small additional labor. We illustrate for the one-state, one-decision variable problem: choose $y(i)$, $i = 0, \ldots, N - 1$, that minimize J given by

$$J = \sum_{i=0}^{N-1} \left[a(i)x^2(i) + c(i)y^2(i) \right], \tag{6.41}$$

$$x(i + 1) = g(i)x(i) + h(i)y(i) \qquad (i = 0, \ldots, N - 1), \tag{6.42}$$

where we wish to solve for all pairs of initial and terminal states, $x(0)$ and $x(N)$.

We define a more general optimal value function by

$F_i(x, z) =$ the minimum value of the criterion for a process
      starting stage i in state x and terminating at stage
      N in state z.

Observe that $F_N(x, z)$ is undefined except when $x = z$, but we can compute $F_{N-1}(x, z)$ by noting that in this case

$$z = g(N - 1)x + h(N - 1)y(N - 1)$$

so

$$y(N - 1) = \frac{z - g(N - 1)x}{h(N - 1)} \tag{6.43}$$

and

$$F_{N-1}(x, z) = a(N - 1)x^2 + c(N - 1)\left[\frac{z - g(N - 1)x}{h(N - 1)} \right]^2$$

$$= \left[a(N - 1) + \frac{c(N - 1)g^2(N - 1)}{h^2(N - 1)} \right]x^2$$

$$+ \left[-\frac{2c(N - 1)g(N - 1)}{h^2(N - 1)} \right]xz + \left[\frac{c(N - 1)}{h^2(N - 1)} \right]z^2. \tag{6.44}$$

For a multidimensional problem with decision dimension less than state dimension, we would use the method of Section 5 to determine F a suitable number of stages before the end. What is important here is that F is a quadratic function jointly in x and z.

The recurrence relation for F is

$$F_i(x, z) = \min_y \left[a(i)x^2 + c(i)y^2 + F_{i+1}(g(i)x + h(i)y, z) \right]$$
$$(i = 0, \ldots, N - 2). \tag{6.45}$$

We assume that $F_{i+1}(x, z)$ has the quadratic form

$$F_{i+1}(x, z) = u(i + 1)x^2 + v(i + 1)xz + w(i + 1)z^2. \tag{6.46}$$

Substitution of this form in (6.45), solution for the minimizing y, and substitution of that y in (6.45) yields (see (6.30) and (6.31) where we have performed this calculation except now z replaces λ)

$$y = - \frac{2u(i + 1)g(i)h(i)x + v(i + 1)h(i)z}{2c(i) + 2u(i + 1)h^2(i)} \tag{6.47}$$

and

$$
\begin{aligned}
F_i(x, z) = {}& \left[a(i) + g^2(i)u(i + 1) - \frac{g^2(i)h^2(i)u^2(i + 1)}{c(i) + u(i + 1)h^2(i)} \right] x^2 \\
&+ \left[g(i)v(i + 1) - \frac{g(i)h^2(i)u(i + 1)v(i + 1)}{c(i) + u(i + 1)h^2(i)} \right] xz \\
&+ \left[w(i + 1) - \frac{h^2(i)v^2(i + 1)}{4c(i) + 4u(i + 1)h^2(i)} \right] z^2 \\
={}& u(i)x^2 + v(i)xz + w(i)z^2. \tag{6.48}
\end{aligned}
$$

Result (6.44) gives us boundary values for $u(N - 1)$, $v(N - 1)$, and $w(N - 1)$ and (6.48) gives us recurrence relations for $u(i)$, $v(i)$, and $w(i)$. Then (6.47) gives us the optimal decision y at each stage for any pair of initial and terminal points.

Problem 6.23. Consider the problem given in Section 3 and delete the $\frac{1}{4} x^2(3)$ term from the criterion. Compute $u(i)$, $v(i)$, and $w(i)$. Use this result to find the optimal solution when $x(0) = 2$, $x(3) = 2$. (Check with the solution to Problem 6.16.) Use the same $u(i)$, $v(i)$, $w(i)$ to find the optimal solution when $x(0) = 4$, $x(3) = -19$.

Problem 6.24. Suppose that in the problem given in section 3 both $x(0)$ and $x(N)$ are unspecified, a term $2x^2(0)$ is added to the cost, and $x(0)$ and $x(N)$ must

be chosen such that

$$x(0) + 2x(N) = 1.$$

Use the calculations from Problem 6.23 to solve this problem.

Problem 6.25. Assume that a, c, g, and h remain constant throughout the process (i.e., do not depend on the stage i). Use the above results to develop a doubling-up procedure. (See Chapter 1, Section 11.)

Chapter 7

DISCRETE-TIME OPTIMAL-CONTROL PROBLEMS

1. Introduction

In this chapter we treat a model similar to, but much more general than, that of the previous chapter. There we treated problems with linear dynamics and quadratic criteria. Our present model has general nonlinear dynamics and general nonquadratic criterion and is called an optimal-control problem. The usual optimal-control problem seeks a continuous function $y(t)$ rather than a discrete sequence $y(i)$, $i = 0, \ldots, N - 1$, so we call the problem of this chapter a discrete-time optimal-control problem. We shall use dynamic programming to develop useful results and computational methods for such problems. The formulas we deduce can also be derived from other viewpoints and are not, themselves, dynamic-programming algorithms. We include them, not only because their derivations represent an interesting use of dynamic programming, but because the results are very useful and represent about all that can be done for the very general problem we consider, particularly if the successive-approximation scheme of the previous chapter diverges.

2. A Necessary Condition for the Simplest Problem

We consider first the following problem with one state and one decision variable: choose $y(i)$, $i = 0, \ldots, N - 1$, that minimize the cost J given by

$$J = \sum_{i=0}^{N-1} g_i(x(i), y(i)) + h(x(N)), \qquad (7.1)$$

95

where

$$x(i + 1) = x(i) + f_i(x(i), y(i)) \qquad (i = 0, \dots, N - 1), \qquad (7.2)$$
$$x(0) = x_0 \qquad \text{(given)}.$$

Here, $x(i)$ and $y(i)$ are continuous variables and f, g, and h are continuous functions with continuous derivatives. Why we choose to write the right-hand side of (7.2) as $x + f$ rather than as f will be explained in Problem 7.3.

We give two derivations of a condition that the optimal solution necessarily satisfies.

Derivation 1. (calculus) Regard the criterion J as a function of the independent variables $x(1), \dots, x(N)$ with $y(0), \dots, y(N - 1)$ considered dependent variables determined by the x's through (7.2). Let x^0 represent the optimal state sequence and y^0 the optimal decision sequence. Then, clearly,

$$\frac{\partial J}{\partial x(i)} \bigg|_{x^0, y^0} = 0, \qquad i = 1, \dots, N.$$

Hence, evaluating $\partial J / \partial x(i)$ for $i = 1, \dots, N$, recognizing that when $x(i)$ is changed, but all other x's are held fixed, $y(i - 1)$ and $y(i)$ must change so as to still satisfy (7.2),

$$\frac{\partial J}{\partial x(i)} = \frac{\partial g_{i-1}}{\partial y(i-1)} \frac{\partial y(i-1)}{\partial x(i)} + \frac{\partial g_i}{\partial x(i)} + \frac{\partial g_i}{\partial y(i)} \frac{\partial y(i)}{\partial x(i)} = 0$$
$$(i = 1, \dots, N - 1), \qquad (7.3)$$

$$\frac{\partial J}{\partial x(N)} = \frac{\partial g_{N-1}}{\partial y(N-1)} \frac{\partial y(N-1)}{\partial x(N)} + \frac{\partial h}{\partial x(N)} = 0,$$

where all terms here and subsequently are evaluated at x^0, y^0.

We obtain formulas for the $\partial y / \partial x$ terms in (7.3) by writing the dynamical equations (7.2) for $i - 1$ and i,

$$x(i) = x(i - 1) + f_{i-1}(x(i - 1), y(i - 1)), \qquad (7.4)$$
$$x(i + 1) = x(i) + f_i(x(i), y(i)), \qquad (7.5)$$

and then taking the partial derivative with respect to $x(i)$, realizing that when doing so all other x's remain fixed while $y(i - 1)$ and $y(i)$ vary dependently. Doing so, (7.4) yields

$$1 = 0 + \frac{\partial f_{i-1}}{\partial y(i-1)} \frac{\partial y(i-1)}{\partial x(i)}$$

so

$$\frac{\partial y(i-1)}{\partial x(i)} = \frac{1}{\partial f_{i-1} / \partial y(i-1)} ; \qquad (7.6)$$

and (7.5) yields

$$0 = 1 + \frac{\partial f_i}{\partial x(i)} + \frac{\partial f_i}{\partial y(i)} \frac{\partial y(i)}{\partial x(i)},$$

so

$$\frac{\partial y(i)}{\partial x(i)} = - \frac{1 + \partial f_i/\partial x(i)}{\partial f_i/\partial y(i)}. \tag{7.7}$$

Necessary conditions (7.3) become

$$\frac{\partial g_{i-1}/\partial y(i-1)}{\partial f_{i-1}/\partial y(i-1)} + \frac{\partial g_i}{\partial x(i)} - \frac{\partial g_i}{\partial y(i)} \frac{1 + \partial f_i/\partial x(i)}{\partial f_i/\partial y(i)} = 0$$
$$(i = 1, \ldots, N - 1), \tag{7.8}$$

and

$$\frac{\partial g_{N-1}/\partial y(N-1)}{\partial f_{N-1}/\partial y(N-1)} + \frac{\partial h}{\partial x(N)} = 0. \tag{7.9}$$

These, plus the N dynamical equations (7.2), represent $2N$ equations for the N unknown x's and N unknown y's, the use of which will be discussed later.

Derivation 2. (dynamic programming) Define the optimal value function by

$S_i(x(i)) = $ the minimum cost of the remaining process if we start stage i in state $x(i)$.

Then, for $i = 0, \ldots, N - 1$,

$$S_i(x(i)) = \min_{y(i)} \left[g_i(x(i), y(i)) + S_{i+1}(x(i) + f_i(x(i), y(i))) \right] \tag{7.10}$$

and

$$S_N(x(N)) = h(x(N)). \tag{7.11}$$

For $y(i)$ to minimize the bracket in (7.10) it is necessary that

$$\frac{\partial g_i}{\partial y(i)} + \frac{\partial S_{i+1}}{\partial x(i+1)} \frac{\partial f_i}{\partial y(i)} = 0 \qquad (i = 0, \ldots, N - 1), \tag{7.12}$$

where the second term results from the chain rule and is the derivative of S with respect to its argument times the derivative of its argument with respect to y. Replacing $y(i)$ in (7.10) by its minimizing value $y^*(i)$ (which is dependent on $x(i)$) and differentiating with respect to $x(i)$ yields

$$\frac{\partial S_i}{\partial x(i)} = \frac{\partial g_i}{\partial x(i)} + \frac{\partial g_i}{\partial y(i)} \frac{\partial y^*(i)}{\partial x(i)} + \frac{\partial S_{i+1}}{\partial x(i+1)} \left(1 + \frac{\partial f_i}{\partial x(i)} \right.$$
$$\left. + \frac{\partial f_i}{\partial y(i)} \frac{\partial y^*(i)}{\partial x(i)} \right) \qquad (i = 0, \ldots, N - 1). \tag{7.13}$$

Result (7.12) causes the sum of the two terms in $\partial y^*(i)/\partial x(i)$ in (7.13) to equal zero, and substitution for $\partial S/\partial x$ in (7.13) using (7.12) reproduces result (7.8). Partial differentiation of (7.11) with respect to $x(N)$ and use of (7.12) with $i = N - 1$ yields (7.9).

3. Discussion of the Necessary Condition

Since we have merely set first derivatives equal to zero (yielding what are called stationary solutions) to obtain (7.8) and (7.9), these conditions are necessary for any relative minimum or relative maximum or even for other types of stationary solutions. A further necessary condition for a relative minimum, easily obtained and used, is that $\partial^2 J/\partial x^2(i) \geqslant 0$, and a stronger condition is that the Hessian matrix

$$\left\| \frac{\partial^2 J}{\partial x(i)\, \partial x(j)} \right\|$$

is nonnegative definite (see any calculus text on the minimization of a function of N variables). Just as with the corresponding calculus problem, there is no way of distinguishing the absolute minimum from other relative minima. This is the principal drawback of this procedure as compared to dynamic programming.

An obvious computational way of using (7.8) and (7.9) follows. (In Section 5 we give a different and more practical procedure.) If we guess a value of $y(0)$, and determine $x(1)$ by (7.2) with $i = 0$, Eq. (7.8) with $i = 1$ becomes an equation in one unknown, $y(1)$. We can solve and then use $y(1)$ in (7.2) with $i = 1$ to get $x(2)$. Then (7.8), for $i = 2$, allows solution for the one unknown, $y(2)$. We can repeat this procedure until (7.8) with $i = N - 1$ gives $y(N - 1)$ and hence $x(N)$. We must adjust our initial guess at $y(0)$ until the solution thus obtained for $y(N - 1)$, $x(N - 1)$, and $x(N)$ satisfies (7.9). Once having done this, as discussed above, we have a candidate for solution, but there is no assurance, in general, that other initial guesses of $y(0)$ will not also satisfy (7.9) and give a smaller value of J.

A similar, alternative, procedure is to guess $\partial S_0/\partial x(0)$ and use (7.12) and (7.13) with $i = 0$ to solve for $\partial S_1/\partial x(1)$ and $y(0)$. Then $x(1)$ is found using (7.2) and the process is repeated with $i = 1$, etc. The values of $\partial S_N/\partial x(N)$ and $x(N)$ thus obtained must satisfy

$$\frac{\partial S_N}{\partial x(N)} = \frac{\partial h}{\partial x(N)}. \tag{7.14}$$

Problem 7.1. How would the above results be modified if instead of having a terminal cost $h(x(N))$ in J we specify the value of $x(N)$? If $x(0)$ were not specified,

but a cost $k(x(0))$ were attached to the choice of $x(0)$ in J, how would the necessary conditions be modified?

Problem 7.2. You have already solved the linear dynamics, quadratic criterion Problem 6.3. Using $\partial V_0/\partial x$ evaluated at $x(0) = 2$ given by that solution as your "guess" of $\partial S_0/\partial x(0)$, verify that (7.12) and (7.13) produce the optimal solution of that problem and that (7.14) is satisfied. Now use $y(0)$ given by the solution of Problem 6.3 to "guess" $y(0)$ and use (7.8) to obtain the same optimal solution and verify that (7.9) is satisfied.

Problem 7.3. Suppose that we write J in (7.1) as

$$J = \sum_{i=0}^{N-1} g_{i\Delta}(x(i\Delta), y(i\Delta))\Delta + h(x(N\Delta))$$

and (7.2) as

$$x((i+1)\Delta) = x(i\Delta) + y(i\Delta)\Delta \qquad (i = 0, \ldots, N-1).$$

(We have added some constants Δ and let $f_i = y$ in the original problem.)

How does this affect results (7.3)–(7.9)? As $\Delta \to 0$ and the number of stages $N \to \infty$ such that $N\Delta = T$ (a constant), the above problem becomes the calculus of variations problem: choose $x(t)$ that minimizes J given by

$$J = \int_0^T g_t(x(t), \dot{x}(t))\ dt + h(x(T))$$

since

$$y(i\Delta) = \lim_{\Delta \to 0} \frac{x((i+1)\Delta) - x(i\Delta)}{\Delta} = \dot{x}(i\Delta).$$

(With this same modification (7.2) becomes, in the limit, $\dot{x} = f$, which is why we separated the x term in (7.2).) What does the new version of (7.8) that you just derived approach as $\Delta \to 0$?

Problem 7.4. Consider the dynamical equation

$$x(i+1) = x(i) + y(i) \qquad (i = 0, \ldots, N-1), \qquad x(0) \text{ and } x(N) \text{ specified},$$

and criterion

$$J = \sum_{i=0}^{N-1} (1 + y^2(i))^{1/2}.$$

This problem requires choosing points $x(i)$ on an i, x coordinate system such that the total length of the straight-line segments connecting consecutive pairs of points is minimal, where the first and last points are specified. Write the necessary conditions (7.8) for this problem. Deduce from them that the solution is the straight line connecting $x(0)$ and $x(N)$.

Problem 7.5. Consider the problem: choose $x(i)$, $i = 0, \ldots, N$, that minimize J given by

$$J = \sum_{i=0}^{N-1} g_i(x(i), x(i+1))$$

subject to the constraint

$$x(0) + x(N) = 1.$$

Use suitable modifications of derivation methods 1 and 2 to deduce in two ways the same conditions necessary for minimality.

Problem 7.6. Consider the problem: choose $y(1), \ldots, y(N - 1)$ that minimize J given by

$$J = \sum_{i=1}^{N-1} g_i(x(i), y(i)) + h(x(N))$$

subject to the dynamical equations

$$x(i + 1) = f_i(x(i), x(i - 1), y(i)) \qquad (i = 1, \ldots, N - 1),$$

where $x(0)$ and $x(1)$ are specified. Use suitable modifications of both derivation methods 1 and 2 to deduce in two ways the same conditions (giving $y(i)$ in terms of $x(i - 3)$, $x(i - 2)$, $x(i - 1)$, $x(i)$, $y(i - 2)$, and $y(i - 1)$ for $i = 3, \ldots, N - 1$) that are necessary for minimality. Discuss what values of y you must guess in order to use these conditions.

4. The Multidimensional Problem

We now consider problem (7.1), (7.2) except that we now regard $x(i)$ as an n-dimensional vector with components $x_j(i)$, $j = 1, \ldots, n$, and $y(i)$ as an m-dimensional vector ($m \leqslant n$) with components $y_j(i)$, $j = 1, \ldots, m$. Hence (7.2) becomes n dynamical equations of the form

$$x_j(i + 1) = x_j(i) + f_{j, i}(x_1(i), \ldots, x_n(i), y_1(i), \ldots, y_m(i))$$
$$(j = 1, \ldots, n; \quad i = 0, \ldots, N - 1). \tag{7.15}$$

If we attempt to modify *derivation* 1 we discover that we want to vary one component of $x(i)$, say $x_k(i)$, while we keep fixed all other components of x at stage i as well as all components of x at all other stages. When performing this variation in $x_k(i)$, all components of $y(i - 1)$ and $y(i)$ are allowed to vary dependently. Then (7.6) becomes n equations for the m unknowns $\partial y_j(i - 1)/\partial x_k(i)$, $j = 1, \ldots, m$, and (7.7) also becomes n equations in m unknowns. These overdetermined equations (unless $m = n$) cannot in general be solved so we must discard this approach.

Derivation 2, the dynamic-programming derivation, is easily modified, fortunately, as follows. Define

$$S_i(x_1(i), \ldots, x_n(i)) = \text{the minimum cost of the remaining}$$
$$\text{process if we start stage } i \text{ in state}$$
$$x_1(i), \ldots, x_n(i).$$

Then, for $i = 0, \ldots, N - 1$,

$S_i(x_1(i), \ldots, x_n(i))$

$$= \min_{y_1(i), \ldots, y_m(i)} \Big[g_i(x_1(i), \ldots, x_n(i), y_1(i), \ldots, y_m(i))$$

$$+ S_{i+1}(x_1(i) + f_{1, i}(x_1(i), \ldots, x_n(i), y_1(i), \ldots, y_m(i)), \ldots, x_n(i)$$

$$+ f_{n, i}(x_1(i), \ldots, x_n(i), y_1(i), \ldots, y_m(i))) \Big] \qquad (7.16)$$

and

$$S_N(x_1(N), \ldots, x_n(N)) = h(x_1(N), \ldots, x_n(N)). \qquad (7.17)$$

Henceforth, we shall not write out in full the arguments of the functions involved.

For $y_1(i), \ldots, y_m(i)$ to minimize the bracketed term in (7.16), the partial derivative with respect to each component must equal zero, so

$$\frac{\partial g_i}{\partial y_k(i)} + \sum_{l=1}^{n} \frac{\partial S_{i+1}}{\partial x_l(i+1)} \frac{\partial f_{l, i}}{\partial y_k(i)} = 0$$

$$(k = 1, \ldots, m; \quad i = 0, \ldots, N - 1). \qquad (7.18)$$

Replacement of $y(i)$ by its minimizing value $y^*(i)$ in (7.16) and partial differentiation with respect to each component of $x(i)$, recognizing the dependence of the minimizing $y(i)$ upon $x(i)$, yields

$$\frac{\partial S_i}{\partial x_k(i)} = \frac{\partial g_i}{\partial x_k(i)} + \sum_{l=1}^{m} \frac{\partial g_i}{\partial y_l(i)} \frac{\partial y_l^*(i)}{\partial x_k(i)} + \frac{\partial S_{i+1}}{\partial x_k(i+1)}$$

$$+ \sum_{l=1}^{n} \frac{\partial S_{i+1}}{\partial x_l(i+1)} \left(\frac{\partial f_{l, i}}{\partial x_k(i)} + \sum_{p=1}^{m} \frac{\partial f_{l, i}}{\partial y_p(i)} \frac{\partial y_p^*(i)}{\partial x_k(i)} \right)$$

$$(k = 1, \ldots, n; \quad i = 0, \ldots, N - 1). \qquad (7.19)$$

The m equations (7.18) cause all the terms in $\partial y^*/\partial x$ in (7.19) to drop out, giving

$$\frac{\partial S_i}{\partial x_k(i)} = \frac{\partial g_i}{\partial x_k(i)} + \frac{\partial S_{i+1}}{\partial x_k(i+1)} + \sum_{l=1}^{n} \frac{\partial S_{i+1}}{\partial x_l(i+1)} \frac{\partial f_{l, i}}{\partial x_k(i)}$$

$$(k = 1, \ldots, n; \quad i = 0, \ldots, N - 1). \qquad (7.20)$$

Finally, partial differentiation of (7.17) yields

$$\frac{\partial S_N}{\partial x_k(N)} = \frac{\partial h}{\partial x_k(N)} \qquad (k = 1, \ldots, n). \qquad (7.21)$$

Results (7.18), (7.20), and (7.21) are necessarily satisfied by the optimal solution, but of course are satisfied by any relative maxima, minima, or other stationary solutions.

To use these results, given $x_k(0)$, $k = 1, \ldots, n$, we might guess the n unknowns $\partial S_0/\partial x_k(0)$. Then (7.18) and (7.20) constitute $n + m$ equations for the $n + m$ unknowns $y_k(0)$, $k = 1, \ldots, m$, and $\partial S_1/\partial x_k(1)$, $k = 1, \ldots, n$. Then (7.15) can be used to find $x_k(1)$, $k = 1, \ldots, n$ and the process repeated $N - 1$ more times. We must guess the n unknowns $\partial S_0/\partial x_k(0)$ in such a way that the resulting $x_k(N)$ and $\partial S_N/\partial x_k(N)$ satisfy the n equations (7.21).

It is important to note that for the general multidimensional problem (with $n \neq m$) there is no way of eliminating the quantities $\partial S/\partial x$ from the statement of the result, as was done in Derivation 2 of Section 2 for the one-dimensional case. In other types of derivations of (7.18), (7.20), and (7.21), what we called $\partial S/\partial x$ appear as Lagrange multipliers, adjoint variables, or costates.

Problem 7.7. From the solution of the linear dynamics, quadratic criterion Problem 6.17, obtain $\partial V_0/\partial x_1(0)$ and $\partial V_0/\partial x_2(0)$ evaluated at $x_1(0) = 7$, $x_2(0) = 1$. Use these as "guesses" of $\partial S_0/\partial x_1(0)$ and $\partial S_0/\partial x_2(0)$ in the appropriate versions of (7.18) and (7.20) and verify that you obtain the correct $y(0)$, $\partial S_1/\partial x_1(1)$, and $\partial S_1/\partial x_2(1)$.

5. The Gradient Method of Numerical Solution

We deduce now a method of computation that turns out, for reasons of numerical stability that we cannot investigate here, to be more efficient than the guessing or shooting procedure described above. We consider here only the free-terminal-point problem (7.1), (7.2) with scalar state and decision, although the procedure can be generalized to multidimensional, fixed-terminal-point problems. We use a method of successive approximation, but one differing from that of the previous chapter. Starting with a guessed decision sequence $y_0(0), \ldots, y_0(N - 1)$ we seek formulas that can be used to find a better (smaller value of J) sequence $y_1(0), \ldots, y_1(N - 1)$. Our derivation uses recursive formulas, but does not use the principle of optimality.

Before presenting the procedure for optimal-control problems, we digress to explain an analogous successive-approximation procedure for the calculus problem: choose x that minimizes $f(x)$ given by

$$f(x) = x^4 - 4x^3 + 8x^2 - 4x + 1.$$

(See Chapter 6, Section 1 where this problem has been discussed.) Suppose we guess that the solution is $x = 0$ (with value $f = 1$). We then evaluate $df/dx|_{x=0}$ and obtain

$$\frac{df}{dx}\bigg|_{x=0} = -4.$$

If we change x by amount δx, to first order f changes by amount δf given by

$$\delta f = \frac{df}{dx}\bigg|_{x=0} \delta x. \tag{7.22}$$

If we decide to change x by an amount proportional to df/dx, so

$$\delta x = k \frac{df}{dx}\bigg|_{x=0} = -4k, \tag{7.23}$$

we obtain

$$\delta f = k\left(\frac{df}{dx}\bigg|_{x=0}\right)^2 = 16k.$$

If we seek a reduction of, say, 1 in the value of f, we get

$$-1 = 16k, \qquad k = -\tfrac{1}{16},$$

so from (7.23), $\delta x = \tfrac{1}{4}$ and our new approximate solution is

$$x_1 = x_0 + \delta x = 0 + \delta x = \tfrac{1}{4}.$$

Evaluating f at $x = \tfrac{1}{4}$ we obtain $113/256$ which is better than the original value $f = 1$ but not as good as the predicted value $f + \delta f = 1 - 1 = 0$. Since it is an improvement on the previous answer, we accept x_1 as our next approximation, compute $df/dx|_{x=1/4}$ and repeat the process.

Note that one can be too greedy with this procedure. If, instead of $\delta f = -1$, we sought $\delta f = -3$, we would have obtained $k = -\tfrac{3}{16}$, $\delta x = \tfrac{3}{4}$, $x_0 + \delta x = \tfrac{3}{4}$, and $f(\tfrac{3}{4}) = 289/256$, which is larger than $f(0)$. We have overshot the minimum and are further away on the other side. If this happens, we would have to seek a smaller reduction, perhaps one-half of the previously sought reduction, i.e., $\delta f = -\tfrac{3}{2}$. If we again overshoot, we would try $\delta f = -\tfrac{3}{4}$, etc.

Figure 7.1 should give the reader a geometric idea of this procedure.

Problem 7.8. Generalize the above procedure to the situation where f is a function of two variables and apply it to minimizing

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$

with initial guess $x_1 = 1$, $x_2 = 1$, and $\delta f = -3$. Apply your formulas, seeking one-half the improvement if the new solution is larger than the old one, until two improvements are achieved.

We return now to the optimal-control problem. Given a guessed decision sequence $\{y_0(i)\}$, $i = 0, \ldots, N - 1$, we define

$T_i(x, y_0(i)) =$ the value of the criterion (7.1) from stage i through N, where we start stage i in state x and use the guessed decision sequence.
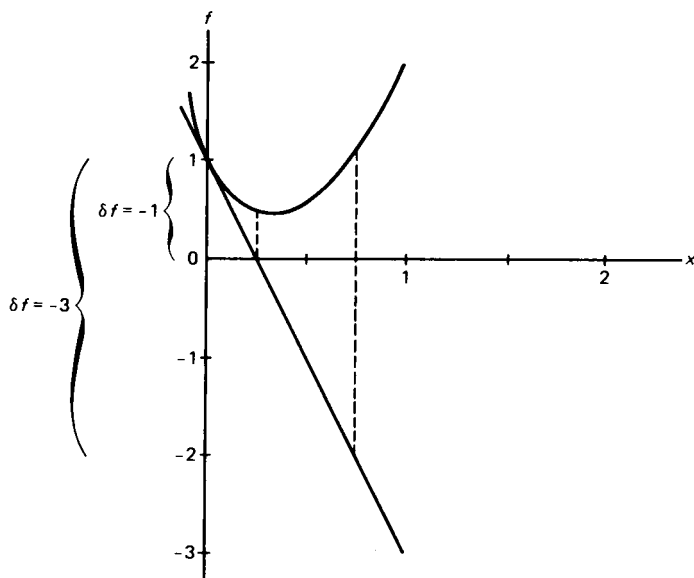
Figure 7.1

We show explicitly the dependence of T on $y_0(i)$ since we shall later allow $y_0(i)$ to vary. T_i is also a function of $y_0(i + 1), \ldots, y_0(N - 1)$.

T_i clearly satisfies the recurrence relation

$$T_i(x, y_0(i)) = g_i(x, y_0(i)) + T_{i+1}(x + f_i(x, y_0(i)), y_0(i + 1))$$
$$(i = 0, \ldots, N - 1) \tag{7.24}$$

and boundary condition

$$T_N(x, \_) = h(x). \tag{7.25}$$

We use (7.2) to compute the state sequence $\{x_0(i)\}$, $i = 1, \ldots, N$, determined by $\{y_0(i)\}$ and $x(0)$. We ask, in order to improve the value of J, how J, hence T_i, would change if we changed $y_0(i)$ but kept all subsequent y's fixed. Partial differentiation of (7.24) with respect to $y_0(i)$ gives the answer:

$$\left.\frac{\partial T_i}{\partial y_0(i)}\right|_0 = \left.\frac{\partial g_i}{\partial y_0(i)}\right|_0 + \left.\frac{\partial T_{i+1}}{\partial x(i + 1)}\right|_0 \left.\frac{\partial f_i}{\partial y_0(i)}\right|_0, \tag{7.26}$$

where $|_0$ means evaluated in terms of the particular sequence $\{y_0(i)\}$, $\{x_0(i)\}$. Hence we could compute $\partial T_i/\partial y_0(i)$ if we knew $\partial T_{i+1}/\partial x(i + 1)$. To find a rule for computing $\partial T_{i+1}/\partial x(i + 1)$ we take the partial derivative of (7.24) with respect to x, obtaining

$$\left.\frac{\partial T_i}{\partial x(i)}\right|_0 = \left.\frac{\partial g_i}{\partial x(i)}\right|_0 + \left.\frac{\partial T_{i+1}}{\partial x(i + 1)}\right|_0 \left(1 + \left.\frac{\partial f_i}{\partial x(i)}\right|_0\right). \tag{7.27}$$

This is a rule for $\partial T/\partial x$ at one stage in terms of $\partial T/\partial x$ at the next stage. From (7.25) we see that

$$\frac{\partial T_N}{\partial x(N)}\bigg|_0 = \frac{\partial h}{\partial x(N)}\bigg|_0 \qquad (7.28)$$

which allows us to compute $\partial T_i/\partial x(i)$ for all i, working backward from stage N.

Formulas (7.26), (7.27), and (7.28) allow the computation of $\partial T_i/\partial y_0(i)$ at each stage along the guessed sequence. We now give a way of using this information to reduce, if possible, the value of J. We decide to change $y_0(i)$, for each stage i, proportionally to the size of $\partial T/\partial y_0(i)$, hence we write

$$\delta y_0(i) = k \frac{\partial T_i}{\partial y_0(i)}\bigg|_0 \qquad (i = 0, \ldots, N - 1). \qquad (7.29)$$

The vector $\partial T/\partial y_0(i)$, $i = 0, \ldots, N - 1$, is called the gradient vector in y space, and (7.29) means that we move in the direction of the gradient. To first order, the resulting change in J, δJ, is approximated by

$$\delta J \approx \sum_{i=0}^{N-1} \frac{\partial T_i}{\partial y_0(i)}\bigg|_0 \delta y_0(i) = k \sum_{i=0}^{N-1} \left(\frac{\partial T_i}{\partial y_0(i)}\bigg|_0\right)^2. \qquad (7.30)$$

(This formula would be exact if J and f in (7.1) and (7.2) were linear functions of their arguments.) For a specified improvement $\overline{\delta J}$ we can use (7.30) to determine k and then (7.29) to find $\{\delta y_0\}$. The new decision sequence $\{y_1(i)\} = \{y_0(i) + \delta y_0(i)\}$, $i = 0, \ldots, N - 1$, is then used in (7.2) to obtain $\{x_1(i)\}$, $i = 1, \ldots, N$, and if the resulting sequences yield a smaller J, they are used as the starting point for another iteration.

If J increases we know that the improvement $\overline{\delta J}$ that we sought was too large and we might replace $\overline{\delta J}$ by $\frac{1}{2}\overline{\delta J}$. Hence, we replace k by $k/2$ in (7.29) and try that solution. As long as we got an increased J we would continue to divide k by 2. We could stop when we seek a reduction δJ of magnitude less than some given small number ϵ and even that is too large and yields a larger J.

Note that no improvement, to first order, is possible when $\partial T_i/\partial y(i) = 0$ for $i = 0, \ldots, N - 1$. In this case, formula (7.26) conforms with necessary condition (7.12), $\partial T_i/\partial x(i)$ equaling $\partial S_i/\partial x(i)$ since (7.27) is the same as (7.13) (with the $\partial y/\partial x$ terms dropped since their coefficient always equals zero) and the boundary condition is the same in both cases.

An extensive comparison of first-order, gradient procedures such as developed in this section and second-order, Newton–Raphson procedures such as given in Problem 6.6 is beyond the scope of this text. Briefly, a gradient procedure requires much less computation per iteration and will converge for initial guesses further from optimal than second-order

procedures. Second-order procedures converge rapidly once close to the solution while gradient procedures encounter difficulties (both the numerator and denominator of the solution for k in (7.30) approach zero) as the solution approaches optimal. A class of methods called conjugate gradient procedures attempts to combine the best features of both of the above schemes.

Problem 7.9. Apply the gradient method to Problem 6.6 with the same initial guess as in Problem 6.6. Let $\delta J = -0.005$ and if the "improved" solution actually is inferior to the previous one, repeatedly divide δJ by 2 until an actual improvement results. Apply the procedure until two actual improvements on the original criterion value have been found.

Problem 7.10. We asserted that (7.30) is exact for linear dynamics and a linear criterion. Let

$$J = \sum_{i=0}^{2} [x(i) + y(i)] + x(3)$$

and

$$x(i + 1) = x(i) + y(i) + 1, \qquad x(0) = 1.$$

For the initial approximation $y(0) = 1$, $y(1) = 1$, $y(2) = 1$, find the x's and compute J. Compute δJ by (7.30), using (7.27) and (7.26). Now, let $y(0) = 1 + \delta y(0)$, $y(1) = 1 + \delta y(1)$ and $y(2) = 1 + \delta y(2)$ and find the x's and evaluate J. Verify that this actual new J exactly equals the original J plus δJ.

Chapter 8

THE CARGO-LOADING PROBLEM

1. Introduction

We come now to a problem with integer-valued variables that is linear in both the dynamics and criterion. This structure permits the development of special computational procedures. Problems of the type that we shall now discuss arise frequently in practice, not merely in the loading of cargo, but also as subproblems in the optimization of such processes as the production of rolled steel (stock cutting) and the expansion of communication networks.

Stated as a cargo-loading problem, we are given the weight capacity of a cargo vehicle and a choice of potential cargo, and seek that cargo of maximum value. Specifically, there are an unlimited number of each of N types of items with each item of type i having weight w_i and value v_i. We are restricted, of course, to an integer number x_i, $x_i = 0, 1, \ldots$, of items of each type i, and therein lies the problem. Were this integer restriction not present, we would merely determine which item type had the largest value-to-weight ratio and load the cargo vehicle full of that type of item. As an illustration, suppose W, the weight capacity of the vehicle, is 9 and that there are four types of items ($N = 4$) with values and weights, per item, as shown in Table 8.1. Since item-type 4 has the largest

Table 8.1. *Data for a hypothetical problem*

Item type i	Weight of item w_i	Value of item v_i	v_i/w_i
1	3	7	$2\frac{1}{3}$
2	6	16	$2\frac{2}{3}$
3	7	19	$2\frac{5}{7}$
4	5	15	3

value-to-weight ratio, the optimal solution, were all x_i not restricted to nonnegative integers, would be $x_4 = \frac{9}{5}$, all other $x_i = 0$. However, as we shall compute later, the optimal integer solution is $x_1 = 1$, $x_2 = 1$, $x_3 = x_4 = 0$. We see that the optimal integer solution is not very similar to the noninteger solution. Rounding the noninteger solution to neighboring integer values will not correctly solve the problem.

Stated mathematically the problem is: given the positive data v_i and w_i, $i = 1, \ldots, N$, and W, choose nonnegative integers x_i, $i = 1, \ldots, N$, that maximize J given by

$$J = \sum_{i=1}^{N} x_i v_i \tag{8.1}$$

subject to

$$\sum_{i=1}^{N} x_i w_i \leqslant W. \tag{8.2}$$

Note that this is a special case of the allocation problem of Chapter 3, Section 1 with the return functions $r_i(x_i)$ being the linear functions $x_i v_i$, and the x_i in constraint (8.2) having coefficients not necessarily 1.

Problem 8.1. How would you solve this problem for integer W and w_i using the standard procedure of Chapter 3?

2. Algorithm 1

We now give the first of a sequence of four algorithms exploiting the linear structure of the problem. Algorithms 3 and 4 to follow are the most efficient, but are somewhat more difficult to explain than the two that we give first.

We assume, for all but Algorithm 4, that (1) the weights w_i, $i = 1, \ldots, N$, are positive integers and that (2) the greatest common divisor of all of the w_i is one. Were assumption (2) not valid (for example, $w_1 = 3$, $w_2 = 9$, $w_3 = 6$, $w_4 = 15$), we could always redefine our unit of weight so as to make it true. (We could take 3 as our unit of weight in the above example so $w_1 = 1$, $w_2 = 3$, $w_3 = 2$, $w_4 = 5$.) If the weights are rational numbers, we can always redefine units so as to make assumption 1 true.

Our objective is to find the optimal solution by recursively computing just one optimal value function rather than the sequence of N such functions as was done in Problem 8.1. We can do this due to the linearity of the functions involved.

Suppose that someone has been loading the cargo vehicle, one item at a time, with each item being of any type he chooses. At some point in the

process he hires you as a consultant to complete the loading process optimally. What information must you elicit from him? Only the remaining unfilled weight capacity is needed. Due to the linearity of the return function we do not care whether he has already loaded ten or zero of, say, item-type 1; if we load one more item its value will be v_1 regardless. Hence, remaining available weight w is the state variable and we define the optimal value function by

$f(w)$ = the maximum value attainable with available weight capacity w.

We decide to load items one at a time so our decision is, "Which item shall we load first?" If we load one of item-type i (we may then decide to load another one of the same type later) we obtain value v_i and have remaining available weight $w - w_i$. Hence, the recurrence relation, for $w = 0, 1, 2, \ldots, W$, is

$$f(w) = \max_{i = 1, \ldots, N} \left[v_i + f(w - w_i) \right]. \tag{8.3}$$

Denoting the smallest w_i by $\underline{w}$, the boundary condition is

$$f(w) = 0 \qquad \text{for} \quad 0 \leqslant w \leqslant \underline{w} - 1 \tag{8.4}$$

and we write, furthermore,

$$f(w) = -\infty \qquad \text{for} \quad w < 0 \tag{8.5}$$

to prohibit loading an item that weighs more than the remaining capacity.

Problem 8.2. For the data given in Table 8.1 of Section 1 and for $W = 22$, use (8.3)–(8.5) to solve the problem. Deduce the optimal set of x_i by keeping track of the optimal policy $p(w)$, where $p(w)$ denotes the maximizing choice of i for available weight w. In case of ties, p will have several values.

We give now a graphical representation of this procedure which will be helpful in understanding Algorithms 3 and 4 that follow. Let us consider a horizontal line representing the available-weight axis and record above the axis the value of $f(w)$ corresponding to each integer w, as shown in Figure 8.1 for $w = 0, 1, \ldots, 10$. Let us also imagine a template made out of N bent-wire arrows, all connected at the point A, where the distance

Figure 8.1

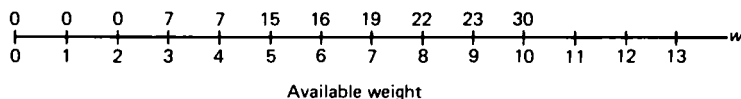between the head of each arrow and A is the weight of item-type i as measured on the scale of Figure 8.1. Such a template is shown in Figure 8.2. Recorded on each bent wire is the value v_i of the item type represented by the wire, as shown in Figure 8.2.

To compute the value of $f(w)$ by Algorithm 1 we place A at available weight w on Figure 8.1. Then, for each item-type i, we add v_i (the number
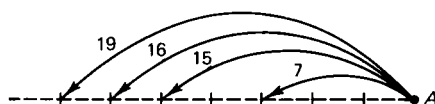
Figure 8.2

on the arrow) to the value of f recorded at the point where the head of the arrow falls (i.e., $w - w_i$). We record at w the largest such sum. The configuration, when we are computing $f(11)$, which equals 31, is shown in Figure 8.3.

Available weight
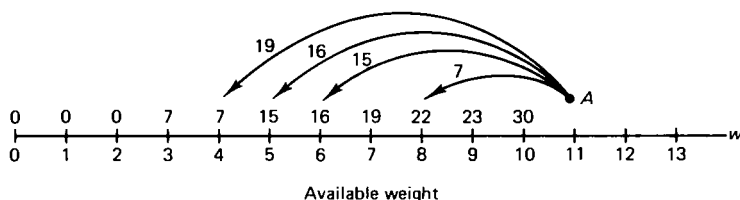
Figure 8.3

3. Algorithm 2

We now make two improvements on Algorithm 1. The first sometimes allows the determination of the optimal solution for capacity W without first computing $f(w)$ for all w less than or equal to W. The second avoids the different representations of the same optimal cargo that occurred in the solution to Problem 8.2. For example, since the optimal solution for weight 14 was either $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$ or $x_1 = x_2 = 0$, $x_3 = 2$, $x_4 = 0$, the first item taken could be either item 1, 2, 3, or 4. Using these different choices of first item, we arrived at the above two distinct final solutions by seven different paths. This can be avoided, without missing out on solutions with the same value but different sets of x_i such as occurred for $w = 14$.

Regarding a way of stopping the computation before w reaches the value W, note first that the item (assumed unique in what follows) having the highest value-to-weight ratio plays a special role. Let us renumber the items if necessary so that this item type is type N. Whenever the available weight w is an integral multiple of w_N, the optimal solution consists only of items of type N since they fit with no slack weight and each gives the best value per unit of weight used. (Check this in Problem 8.2 for $w = 5$, 10, 15, and 20.) It is only when w is such that item N cannot be used without some slack that other items may enter the optimal solution, possibly completely replacing item-type N, such as for $w = 9$ in Problem 8.2.

Consider Figure 8.4. Letting N be the item type of largest value-to-weight ratio, for any data the dotted line of slope v_N/w_N always lies above or on the plot of the optimal value function since for all w it corresponds to the value of the cargo consisting of w/w_N items of type N, even if w/w_N is not an integer and therefore not an admissible cargo. (As argued above, if w is such that w/w_N is an integer, the optimal value function intersects the dotted line.) The solid line lies below or on the plot of the optimal value function since it corresponds to a feasible but not necessarily optimal cargo consisting of $[w/w_N]$ (recall that this notation denotes the greatest integer less than or equal to w/w_N) items of type N with the remaining weight unused. Hence the optimal value function lies between the dotted and solid lines. Let item-type $N-1$ have the second-highest value-to-weight ratio. The dashed line of slope v_{N-1}/w_{N-1} lies above or on a plot of the optimal value function for a modified problem p' in which item-type N is not allowed in the cargo. Clearly there exists a w' such that for all $w \geqslant w'$ the best possible cargo for the modified problem p' has value less than the feasible value (solid line) for the actual problem, hence value less than the optimal value for the actual problem. Since any solution of value greater than the dashed line must include item-type N, we have shown that for all available weights $w \geqslant w'$, item-type N is contained in the optimal cargo.

Problem 8.3.   The line of alternating dashes and dots of Figure 8.4 clearly lies below the plot of the optimal value function. Write its equation and that of the dashed line and solve for w^*, the value of w where they intersect. w' as defined
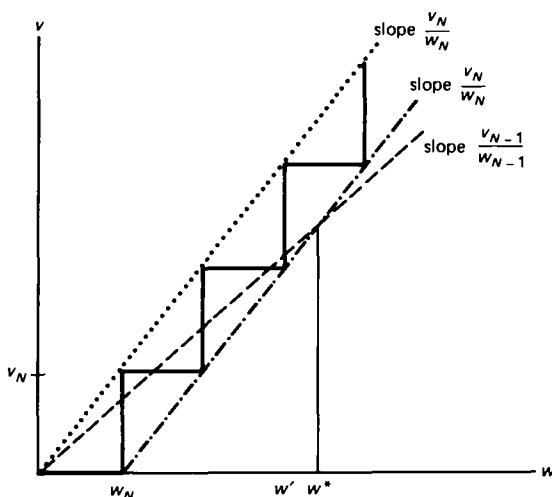

Figure 8.4

above equals $[w^*/w_N]w_N$. Compute w' for the data of Table 8.1. Since the true optimal value function never lies below and generally lies above the solid line, the smallest weight capacity $\tilde{w}$ such that for $w \geqslant \tilde{w}$ item-type N is always in the optimal cargo will generally be considerably less than w'.

Letting $\bar{w}$ denote the largest w_i, we can recognize the occurrence of the event that item-type N is in all optimal cargos for all $w \geqslant \tilde{w}$ when, in Algorithm 1, item-type N is part of every optimal solution for each of $\bar{w}$ consecutive values of w. (This first occurs in our example for the seven consecutive values $w = 15, 16, \ldots, 21$ so $\tilde{w}$ is 15 although w' of Problem 8.3 was 50.) Let z denote the largest weight of the $\bar{w}$ consecutive weights having item-type N in all of their optimal solutions. In terms of the template of Figure 8.2, when point A is placed at $z + 1$, the head of each arrow will point to a weight for which item-type N is in every optimal solution. Hence, item-type N is in every optimal solution for weight $z + 1$. The same argument then applies at weight $z + 2$, $z + 3$, $\ldots$. Hence we can stop the computation after finding $\bar{w}$ consecutive weights with item-type N in every optimal solution. The optimal solution for any weight w larger than z is then found by first determining the smallest integer k such that $w - kw_N$ is less than or equal to $z - \bar{w}$. Take k items of type N and then use the optimal policy table starting from weight $w - kw_N$ to complete the optimal solution. (If we desired only *an* optimal solution in case of ties, we could stop the algorithm after finding $\bar{w}$ consecutive weights having item-type N in *an* optimal solution. In our example we would stop at $w = 16$ rather than $w = 21$.)

Problem 8.4. What is the optimal solution to Problem 8.2 for $W = 99$? For $W = 100$?

Now we give a procedure for avoiding different representations of the same optimal solution. We stipulate that we are going to load items only in such an order that the item-type designation is nonincreasing. For example, if we want to load one of item-type 1, three of type 2, and two of type 4, in order to achieve nonincreasing designations we must load one item of type 4, then the second item of type 4, then the three items of type 2 (one at a time) and then the item of type 1. Should we try any other order, like one of the items of type 2 after the first item of type 4, we could never load the second item of type 4 since that would cause an increase from 1 or 2 to 4 in the item-type designation. Obviously, each unique set of x_i has one and only one such nonincreasing enumeration.

To limit our consideration to such representations, we modify Algorithm 1 as follows. Let the optimal policy function $q(w)$ equal the type of the first item loaded under the above enumeration. If the optimal solution is unique, so will be $q(w)$. If there is more than one optimal

solution with differing first items (such as for $w = 14$ in our example), $q(w)$ will be multivalued. Now, when you consider loading an item of type i when available weight is w, first compute $w - w_i$ and examine $q(w - w_i)$ or the smallest value of $q(w - w_i)$ if it is not unique. If the examined $q(w - w_i)$ is greater than i, this is not a nonincreasing enumeration. Do not consider that particular i further and proceed to the next type of item.

Problem 8.5.   Use Algorithm 2 to solve the problem given by Table 8.1. Keep track of $q(w)$ and use it to deduce the optimal cargo for each value of w. Stop after $w = 21$ due to the stopping rule given above.

4.   Algorithm 3

We call Algorithms 1 and 2 *backward-looking* procedures because we compute a particular $f(w)$ by looking back at previously computed $f(w)$'s. In terms of Figure 8.2, the heads of the arrows are to the left of A. We give now a *forward-looking* procedure in which we turn the template of Figure 8.2 around. When the permanent (optimal) value of $f(w)$ is established for some w, we use this to compute temporary (obtainable but perhaps not optimal) values of $f(w)$ for larger values of w. Another example, with which the reader is already familiar, of computing temporary values which only later become permanent is the Dijkstra procedure for finding shortest paths given in Chapter 4, Section 3A.

The reader should recognize as we describe the procedure that, except for a small exception to be noted later, we are doing exactly the same computations as in Algorithm 2, only we are doing them in a somewhat different order.

The key idea is this. As soon as we know the correct value of $f(w)$ and the optimal first item $q(w)$ for some particular w, we also know, for any i, that an attainable value for $f(w + w_i)$ is $v_i + f(w)$ since this would be the weight and value if we add one item of type i to the optimal cargo for weight w. Furthermore, we know that if we restrict ourselves to nonincreasing enumerations of solutions, when we know $f(w)$ and $q(w)$, we need consider only $v_i + f(w)$ as a candidate for $f(w + w_i)$ for i greater than or equal to $q(w)$. (In case $q(w)$ is not unique, consider i greater than or equal to the minimum $q(w)$.) After computing a temporary value for $f(w + w_i)$ as described, we compare it to the best temporary value of $f(w + w_i)$ already computed, if any, and accept the new temporary value only if it equals or exceeds the old temporary value. If we accept it, we record i as the temporary value of $q(w + w_i)$ that goes with the new temporary $f(w + w_i)$. If we start with $w = 0$, increase w by one at each step, and use this procedure for computing temporary values for larger w,

it is clear that when $w = x$, the temporary value of $f(x)$ is the optimal value since no further steps will affect it.

We now give Algorithm 3A for this procedure. Recall that $\underline{w}$ denotes the smallest w_i and $\overline{w}$ denotes the largest. Let $f^0(w)$ and the set $q^0(w)$ denote temporary values of the optimal value function and the optimal policy, respectively. Let $f(w)$ and $q(w)$ denote permanent values.

Step 0:   Set $f^0(w) = 0$, $q^0(w) = \{0\}$ for $w = 0, 1, 2, \ldots, \underline{w} - 1$; $f^0(w)$ $= -\infty$ for $w = \underline{w}, \underline{w} + 1, \ldots$. Set $w = 0$.

Step 1: $f(w) = f^0(w)$, $q(w) = q^0(w)$.

Step 2:   For each $i \geqslant \min q(w)$, replace $f^0(w + w_i)$ by $\max[v_i + f(w), f^0(w + w_i)]$. If the former is larger, replace $q^0(w + w_i)$ by $\{i\}$. In case of a tie, replace $q^0(w + w_i)$ by $q^0(w + w_i) \cup \{i\}$.

Step 3:   If there exists a nonnegative integer z such that $q(w - z) = \{N\}, \ldots, q(w) = \{N\}$, $q^0(w + 1) = \{N\}, \ldots, q^0(w - z + \overline{w} - 1) = \{N\}$, stop. If not, increase w by 1 and go to step 1.

Problem 8.6.   Justify step 3 of the algorithm.

Problem 8.7.   Compute $f(w)$ for $w = 0, 1, \ldots, 16$ by Algorithm 3A using the data of Table 8.1. This should duplicate the result of Problem 8.5.

Thus far we have done exactly the same computations as in Algorithm 2 only in a different order. In terms of the template of Figure 8.2, we can depict the procedure in Figure 8.5. The computational efficiency stems from the fact that in Algorithm 2 we had to compute $w - w_i$ for each i, $i = 1, \ldots, N$, in order to find out if $i \geqslant \min q(w - w_i)$, in which case we computed the value. In Algorithm 3A we know $q(w)$ when we do step 2 so we can immediately compute temporary values for all $i \geqslant \min q(w)$.

We now present Algorithm 3B, a further refinement of Algorithm 3A which slightly improves its efficiency and which becomes the basis of Algorithm 4 to follow. This algorithm is due to Gilmore and Gomory [2]. Note that sometimes adding one to the available weight does not affect the solution. The additional weight does not allow an improved combination of items so we merely do not use all of the available weight. This was the case in going from $w = 3$ to $w = 4$ in our example. We call *breakpoints* those values of w for which a better solution than for $w - 1$ exists or for
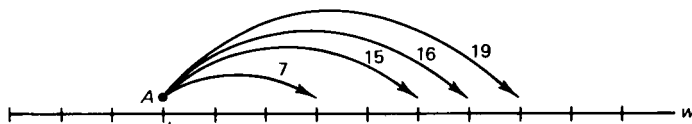

Figure 8.5

which a different solution of the same value exists. Clearly in the backward procedure of Algorithm 2 a weight w can be a breakpoint only if for some i, $w - w_i$ is a breakpoint. Similarly, in the forward Algorithm 3A if we place A in Figure 8.5 only at breakpoints, we shall compute temporary values only at points that may be breakpoints. These are the only points that need concern us since nonbreakpoint values of $f(w)$ are by definition the same as $f(w - 1)$. Hence we need apply step 2 only when $f(w) > f(w - 1)$.

Algorithm 3B is as follows:

Step 0: Set $f(-1) = -\infty$. Set $f^0(0) = 0$, $q^0(0) = \{0\}$. Set $f^0(w) = -\infty$ for $w = 1, 2, \ldots$. Set $w = 0$.

Step 1: $f(w) = \max [f(w - 1), f^0(w)]$. If $f(w - 1)$ yields the maximum, $q(w) = q(w - 1)$ and go to step 3. If $f^0(w)$ is larger, $q(w) = q^0(w)$. In case of a tie, $q(w) = q^0(w) \cup q(w - 1)$.

Step 2: Same as step 2 of Algorithm 3A.

Step 3: Same as step 3 of Algorithm 3A.

Problem 8.8. Compute $f(w)$ for $w = 0, 1, \ldots, 5$ for the data of Table 8.1 using the above modification and check that you get the same result as in Problem 8.7.

5. Algorithm 4

We now give a backward-looking procedure due to Dreyfus and Prather [1] for problems where the greatest common divisor of the w_i is very small relative to W. As an example, suppose we modify Table 8.1 by adding 0.001 to each w_i, obtaining $w_1 = 3.001$, $w_2 = 6.001$, $w_3 = 7.001$, and $w_4 = 5.001$, and that we let $W = 100$. The greatest common divisor of these data is $1/1000$ and division by this gives $w_1 = 3001$, $w_2 = 6001$, $w_3 = 7001$, $w_4 = 5001$, $W = 100,000$. For these data it turns out that the application of Algorithm 2 would require computation of about 27,000 values of $f(w)$ before $q(w)$ equaled N (i.e., 4) for $\bar{w}$ (i.e., 7001) consecutive values. At all but about 30 of these 27,000 points $f(w)$ would equal $f(w - 1)$. Even Algorithm 3B would be repeated 27,000 times, although step 2 would be skipped most of the time.

By focusing on breakpoints as introduced in Algorithm 3B, versions of Algorithm 2 or 3B can be stated that are insensitive to the greatest common divisor of the w_i. The algorithm that we now describe is based on Algorithm 2, and focuses attention only on possible breakpoints.

We have already stated that $f(w)$ can have a breakpoint only at

weights w such that for some i, $w - w_i$ is a breakpoint. Examine the template and w line shown in Figure 8.3 (except that each bent arrow is lengthened by 0.001 between head and tail for our new data) and assume that f has just been computed for a particular w, call it u, which is a breakpoint. We now know all breakpoints of f occurring at weights less than or equal to u. To find the next larger value of w where f *might* experience a breakpoint we slide the template to the right along the w axis until A is at an available weight, call it z, such that the head of one or more arrows coincides with breakpoints with associated policies q (or min q in case q is not unique) less than or equal to the item type represented by each arrow. If there is one such arrow (let it correspond to item-type k), compare $v_k + f(z - w_k)$ to $f(u)$. If the former is larger, z is a breakpoint and we set $f(z) = v_k + f(z - w_k)$ and $q(z) = k$. If $f(u)$ is larger, we continue to slide the template to the right until another coincidence of some arrowhead and previous breakpoint occurs. If $f(u) = v_k + f(z - w_k)$, we declare z a breakpoint and adjoin k to the optimal policy at u to obtain $q(z)$. If k is not unique, we compare the largest $v_k + f(z - w_k)$ to $f(u)$.

Let us carry this out for our modified standard example. We regard $w = 0$ as breakpoint number 0 with $f(0) = 0$ and $q(0) = 0$. At $w = 3.001$ the head of the shortest arrow (item-type 1) is at $w = 0$, so we compare $v_1 + f(0)$ to $f(0)$; and since the former is larger, we record $w = 3.001$ as breakpoint number 1 with $f(3.001) = 7$ and $q(3.001) = 1$. Letting w increase, at $w = 5.001$ the second shortest arrow ($i = 4$) has its head at 0, and we compute that $w = 5.001$ is breakpoint number 2 with $f(5.001) = 15$, $q(5.001) = 4$. At $w = 6.001$ we get breakpoint number 3 with $f(6.001) = 16$, $q(6.001) = 2$. At $w = 6.002$ the head of the $i = 1$ arrow is at breakpoint number 1, but since $7 + 7 < 16$ this is not a breakpoint and we continue to increase w. At 7.001 we obtain breakpoint number 4 with $f(7.001) = 19$ and $q(7.001) = 3$. At 8.002 the head of the arrow for item-type 4 is at 3.001 and f jumps to 22, breakpoint number 5 is $f(8.002) = 22$, $q(8.002) = 4$. At 9.002, f jumps to 23, q is 2, and we have breakpoint number 6. Note that $w = 9.003$ is not a potential breakpoint although it corresponds to three items of type 1, because 6.002 was not a breakpoint. At 10.002, the arrow of item-type 3 points to 3.001 and gives 26; and the arrow of item-type 4 points to 5.001 and gives 30, so this is breakpoint number 7 with $f(10.002) = 30$, $q(10.002) = 4$.

We hope that the idea behind the procedure is now clear, but we are sure that the reader is having some trouble figuring out, at each step, where the next potential breakpoint occurs. The easiest way to do this is to keep track, separately for each item-type i, of the next value of w for which its arrow points to a breakpoint with $q \leqslant i$ and the number, call it p, of that breakpoint. Hence there is one value of "next interesting w" for each

item-type i, and the minimum of these values (let it correspond to item-type j) is the next potential breakpoint. After examining that point, item-type-j's "next interesting w" is increased to w_j plus the weight of breakpoint $p + 1$ if the minimum q at breakpoint $p + 1$ is less than or equal to j. If $q > j$, try w_j plus the weight of breakpoint $p + 2$, etc. If, at any point, no breakpoint $p, p + 1, \ldots$, up to the last one computed has minimum q less than or equal to j, item-type j is dropped from all future consideration. Eventually every item type except N will be dropped and the computation terminates.

Problem 8.9. Organize the computation of the modified standard example based on the above remarks and complete the solution.

We now give several general problems concerning the formulation and solution of cargo-loading problems.

Problem 8.10. Develop a modification of Algorithm 2 that solves the cargo-loading problem where each item-type i is characterized by a positive integer weight w_i, a positive integer volume u_i, and a value v_i, and there are constraints on both available weight and available volume.

Problem 8.11. Consider the problem:

$$\max_{\{x_i\}} \sum_{i=1}^{N} x_i v_i$$

$$\text{subject to} \quad \sum_{i=1}^{N} x_i w_i \leqslant W \qquad \text{(all } w_i \text{ are positive integers),}$$

$$x_i = 0 \text{ or } 1 \qquad (i = 1, \ldots, N).$$

This is sometimes called the knapsack problem, and its solution can be used to decide what items to carry on a backpacking trip. Give an efficient dynamic-programming solution procedure.

Problem 8.12. Suppose in Problem 8.11 that you have two knapsacks available, one with capacity $\overline{W}$ and the other with capacity $\overline{\overline{W}}$. Only one item of each of N types is available and you want to load the two knapsacks so as to maximize the sum of their values. Give an efficient dynamic-programming solution procedure.

Problem 8.13. Consider the following algorithm, involving the recursive computation of only one function, for the knapsack problem given in Problem 8.11:

$$f(w) = -\infty \qquad \text{for} \quad w < 0,$$

$$f(w) = 0 \qquad \text{for} \quad w = 0, 1, \ldots, \underline{w} - 1 \; (\underline{w} \text{ is the minimum } w_i),$$

$$f(w) = \max \left\{ \begin{array}{l} \max_{(P)} [v_i + f(w - w_i)] \\ f(w - 1) \end{array} \right\} \qquad \text{for} \quad w = \underline{w}, \underline{w} + 1, \ldots,$$

where $\{P\}$ is the set of all i, $i = 1, \ldots, N$, such that an optimal solution for weight $w - w_i$ does not contain item-type i.

Either prove this procedure correct or show it to be incorrect by constructing a counterexample in which it yields a less than optimal value of $f(w)$ for some w.

Problem 8.14. Suppose that you have solved (say by Algorithm 3B) a cargo-loading problem with N types of items. Then you are told that z items of an $(N + 1)$th type are also available, with positive integer weight per item w_{N+1} and value per item v_{N+1}. Give an efficient procedure for solving the problem with $N + 1$ types of items using your solution of the problem with N types of items.

References

[1] Dreyfus, S. E., and K. L. Prather, "Improved Algorithms for Knapsack Problems," Report ORC 70-30, Operations Research Center, Univ. California, Berkeley (1970).

[2] Gilmore, P. C., and R. Gomory, "The Theory and Computation of Knapsack Functions," *Operations Res.* **14**, (1966), 1045–1074.

Chapter 9

STOCHASTIC PATH PROBLEMS

1. Introduction

We have assumed thus far in this text that the cost and the change in the state resulting from every decision, even those far in the future, are known with certainty. While this has an appealing simplicity and such a world could be managed quite efficiently, this is the case neither in our own personal lives nor in the business and industrial worlds. We turn now to a somewhat closer approximation to reality and assume that each decision can result in a number of possible outcomes, each with a known probability.

An even more realistic model, the topic of Chapter 15, assumes that even the probabilities of outcomes are not known to us, but that we learn something about them from experience as the process evolves. The problems treated in Chapter 15 are of necessity quite simple due to the excessive computational requirements of models involving learning. Since artful operations research tempers realistic complexity with computational feasibility, the stochastic models of the type treated in this portion of the text, which compromise between certainty and learning, are extremely important and fruitful. Practical applications will be given in the next chapter. Here, we use elementary path problems to develop and illustrate important concepts.

Perhaps the principal attraction of dynamic programming is the almost trivial ease with which it can be adapted, both mathematically and computationally, to stochastic situations. No other technique of operations research can make a comparable claim. As will become clear later, this ease stems from the necessity in deterministic dynamic programming of solving a large number of subproblems in order to solve a given whole problem, a property neither shared nor envied by other techniques.

2. A Simple Problem

We return to a one-way city of the type from which we started our journey in Chapter 1. A map of the city, together with the costs of various arcs, is shown in Figure 9.1. We imagine that we have been hired as consultant to a forgetful traveler who wishes to get from A to line B at minimum cost. Our problem is compounded by the fact that if we instruct the traveler to go diagonally up (a decision we designate U) he remembers our advice and does so with probability $\frac{3}{4}$; with probability $\frac{1}{4}$ he forgets and does the opposite, taking the diagonally downward arc. Likewise, if our instruction is to move diagonally downward (decision D), he complies with probability $\frac{3}{4}$, but moves upward instead with probability $\frac{1}{4}$. He behaves this way at each vertex, no matter how good or bad his prior memory. Consequently, no matter what our instructions, we cannot be sure of the path our employer will follow, but our advice will certainly determine the probabilities of various results. We wish to minimize the expected cost of the trip, assuring us that if the absentminded traveler repeats his journey a great many times, encountering different costs on different trips due to different memory lapses (if any), the average cost will be minimized. (An alternative criterion that we might adopt, among others, would be to maximize the probability that the path our traveler follows costs less than some number Z, the amount that he can afford. This criterion will be the subject of an assigned problem.)
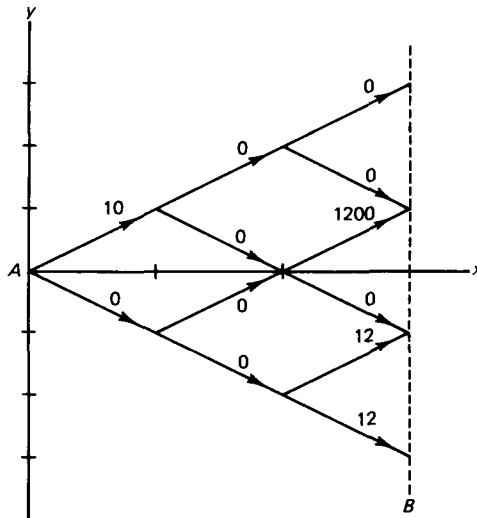
Figure 9.1

3. What Constitutes a Solution?

The question now arises, What form should our solution take? In Chapter 1 we used dynamic programming to determine first the optimal *policy* function giving a decision for every vertex, and then deduced from it the actual optimal *sequence* of decisions for initial vertex *A*. The policy and the sequence were two representations of the same optimal solution. The former also solved other problems and the latter was more compact and simple to communicate.

For our uncertain (stochastic) problem, a policy and a sequence are quite different matters since a sequence of decisions dictates the second and subsequent decisions independently of the outcome of previous decisions, while a policy is outcome dependent. For example, the (perhaps nonoptimal) solution *sequence* "go up, then down, then up" means that our traveler will try to remember, at step 2, to go down (thereby doing so with probability $\frac{3}{4}$) whether he correctly executed our first instruction ("go up") or not. If our *policy* says "go up at step 1 and then if you are at (1, 1), i.e., you indeed went up at the first step, then go down, but if you are at (1, − 1), having forgotten our first instruction, then go up," then what he tries to remember to do will be different for different actual situations.

Whether we present the solution as a policy or a sequence depends entirely on which we are hired to do. The optimal sequence is easier to state and does not require that the traveler observe where he is at each stage (i.e., observe the state, or *y* coordinate). However, the optimal policy will always yield an average cost at least as small as, and usually smaller than, the optimal sequence since it allows more flexibility. The optimal sequence can be thought of as an optimal policy restricted by the stipulation that all decisions at a given stage must be the same, no matter what the state.

In conformity with control engineering terminology, we shall call the solution specified by a sequence of decisions *open-loop* control and the solution specified by a policy *feedback* control. Dynamic programming yields the optimal feedback control, as we shall see below.

4. Numerical Solutions of Our Example

To determine the best open-loop control sequence for the problem of Figure 9.1, we consider all eight possible sequences of three decisions each, and choose the one with minimum expected cost. For example, the decision sequence *D–U–D* that optimizes the deterministic version of this problem has probability 27/64 of actually yielding the path consisting of a

downward, then an upward, and finally a downward transition. This path has cost 0. There is a probability of 9/64 of an upward–upward–downward path of cost 10. Unfortunately, there is also a probability of 9/64 of a downward–upward–upward path costing 1200, etc. Multiplying each of the eight appropriate costs by their respective probabilities and adding, we obtain an expected cost E_{DUD} which is given by

$$E_{DUD} = \tfrac{27}{64} \cdot 0 + \tfrac{9}{64}(10 + 12 + 1200) + \tfrac{3}{64}(12 + 10 + 10) + \tfrac{1}{64} \cdot 1210 = 192\tfrac{1}{4}.$$

It turns out that the decision sequence U–U–D has the minimum expected cost, $120\tfrac{3}{16}$.

The optimal feedback control may be computed recursively just as in the deterministic examples. We begin by defining the optimal expected value function (note the adjective "expected") by

> $S(x, y) = $ the expected cost of the remaining process if we start at vertex (x, y) and use an optimal feedback control policy.

Then, if we choose decision U at (x, y), with probability $\tfrac{3}{4}$ we make a transition to $(x + 1, y + 1)$, with the first-step cost of $a_u(x, y)$ and remaining expected cost $S(x + 1, y + 1)$. With probability $\tfrac{1}{4}$ we make a transition to $(x + 1, y - 1)$, at a first-step cost $a_d(x, y)$ and with remaining expected cost $S(x + 1, y - 1)$. The probabilities are reversed for initial decision D. Consequently, by a stochastic version of the principle of optimality, to be rigorously justified below, we have

$$S(x, y) = \min \left[\begin{array}{l} U: \quad \tfrac{3}{4}\{a_u(x, y) + S(x + 1, y + 1)\} \\ \qquad + \tfrac{1}{4}\{a_d(x, y) + S(x + 1, y - 1)\} \\ D: \quad \tfrac{1}{4}\{a_u(x, y) + S(x + 1, y + 1)\} \\ \qquad + \tfrac{3}{4}\{a_d(x, y) + S(x + 1, y - 1)\} \end{array} \right]. \quad (9.1)$$

The boundary condition is

$$S(3, 3) = 0, \qquad S(3, 1) = 0, \qquad S(3, -1) = 0, \qquad S(3, -3) = 0. \quad (9.2)$$

Use of (9.1) and (9.2) for the problem of Figure 9.1 yields Figure 9.2, where the values of the optimal expected value function are shown in circles and the optimal decisions (but not necessarily the transition which results, which is a random event) are given in squares. The expected cost using the optimal feedback control policy is $84\tfrac{1}{4}$.

The optimal feedback control policy is that set of decisions, one for each stage and state, that minimizes the expected cost starting from $(0, 0)$, which we denote by $V(0, 0)$. The expected cost $V(0, 0)$ is the sum over all paths of the product of the probability of each path times the cost of the
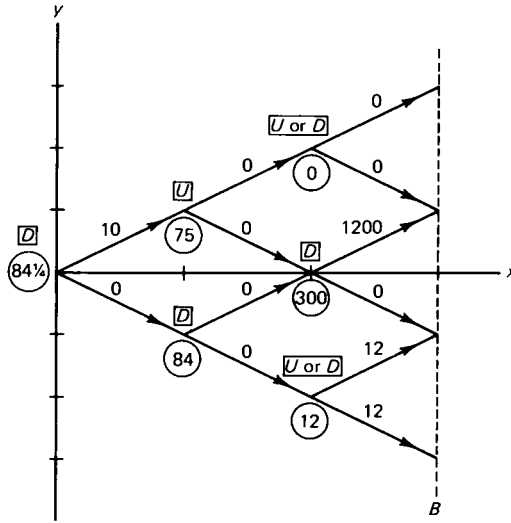
Figure 9.2

path. We now show that the cost $S(0, 0)$ as computed recursively by (9.1) and (9.2) is indeed the minimum expected cost $V(0, 0)$ and, hence, that the principle of optimality is indeed valid for stochastic problems. Then, rather than repeat this type of proof in future applications, we shall usually merely cite the principle.

Let us define $p(i, j)$ as the probability of going diagonally upward at vertex (i, j) and $q(i, j) = 1 - p(i, j)$. We are to choose $p(i, j)$ to be either $\frac{3}{4}$ or $\frac{1}{4}$ at each of six vertices. By definition,

$$
\begin{aligned}
V(0, 0) = \min_{\substack{p(0,0),\, p(1,1),\\ p(1,-1),\, p(2,2),\\ p(2,0),\, p(2,-2)}} &[\, p(0, 0)p(1, 1)p(2, 2)\{ a_u(0, 0) + a_u(1, 1) + a_u(2, 2)\} \\
&+ p(0, 0)p(1, 1)q(2, 2)\{ a_u(0, 0) + a_u(1, 1) + a_d(2, 2)\} \\
&+ p(0, 0)q(1, 1)p(2, 0)\{ a_u(0, 0) + a_d(1, 1) + a_u(2, 0)\} \\
&+ p(0, 0)q(1, 1)q(2, 0)\{ a_u(0, 0) + a_d(1, 1) + a_d(2, 0)\} \\
&+ q(0, 0)p(1, -1)p(2, 0)\{ a_d(0, 0) + a_u(1, -1) + a_u(2, 0)\} \\
&+ q(0, 0)p(1, -1)q(2, 0)\{ a_d(0, 0) + a_u(1, -1) + a_d(2, 0)\} \\
&+ q(0, 0)q(1, -1)p(2, -2)\{ a_d(0, 0) + a_d(1, -1) + a_u(2, -2)\} \\
&+ q(0, 0)q(1, -1)q(2, -2)\{ a_d(0, 0) + a_d(1, -1) + a_d(2, -2)\} \,].
\end{aligned}
$$

Regrouping terms and writing the minimization in front of only those terms that involve the minimized variable,

$$V(0, 0) = \min_{p(0, 0)} \left[p(0, 0)a_u(0, 0) + q(0, 0)a_d(0, 0) \right.$$

$$+ \min_{p(1, 1), p(1, -1)} \left\{ p(0, 0)p(1, 1)a_u(1, 1) + p(0, 0)q(1, 1)a_d(1, 1) \right.$$

$$+ q(0, 0)p(1, -1)a_u(1, -1) + q(0, 0)q(1, -1)a_d(1, -1)$$

$$+ \min_{p(2, 2)} \left[p(0, 0)p(1, 1)p(2, 2)a_u(2, 2) \right.$$

$$+ p(0, 0)p(1, 1)q(2, 2)a_d(2, 2)]$$

$$+ \min_{p(2, 0)} \left[\{ p(0, 0)q(1, 1) + q(0, 0)p(1, -1) \}p(2, 0)a_u(2, 0) \right.$$

$$+ \{ p(0, 0)q(1, 1) + q(0, 0)p(1, -1) \}q(2, 0)a_d(2, 0)]$$

$$+ \min_{p(2, -2)} \left[q(0, 0)q(1, -1)p(2, -2)a_u(2, -2) \right.$$

$$\left. \left. + q(0, 0)q(1, -1)q(2, -2)a_d(2, -2)] \right\} \right].$$

Regrouping and substituting S given by (9.1) and (9.2),

$$V(0, 0) = \min_{p(0, 0)} \left[p(0, 0)a_u(0, 0) + q(0, 0)a_d(0, 0) \right.$$

$$+ \min_{p(1, 1)} \left\{ p(0, 0)p(1, 1)[a_u(1, 1) + S(2, 2)] \right.$$

$$+ p(0, 0)q(1, 1)[a_d(1, 1) + S(2, 0)]\}$$

$$+ \min_{p(1, -1)} \left\{ q(0, 0)p(1, -1)[a_u(1, -1) + S(2, 0)] \right.$$

$$\left. \left. + q(0, 0)q(1, -1)[a_d(1, -1) + S(2, -2)]\} \right\} \right]$$

$$= \min_{p(0, 0)} \left[p(0, 0)\{a_u(0, 0) + S(1, 1)\} \right.$$

$$\left. + q(0, 0)\{a_d(0, 0) + S(1, -1)\} \right]$$

$$= S(0, 0).$$

This completes the proof of the validity of the principle of optimality for this problem. The reader can now see why in the future we prefer to cite an intuitively obvious principle rather than construct a rigorous proof.

5. A Third Control Philosophy

At this point, we would like to introduce a third control scheme. Let us use the optimal open-loop solution to yield our initial decision for the three-stage problem. Then, after a transition has ocurred, let us observe the result and determine the best open-loop solution for the new two-stage problem. After implementing the initial control decision of this optimal

open-loop solution, a transition occurs and we again observe the state and use the optimal control decision for the remaining one-stage problem. This scheme uses the optimal open-loop initial decision at each stage, but incorporates feedback in that it solves problems starting in the actual current state. We call this scheme *open-loop-optimal feedback* control.

For the example problem of Section 2, this control scheme differs from both of the previous schemes. It turns out that the open-loop-optimal feedback decision coincides with the optimal feedback decision at each vertex, except for vertex A. There, as has been shown, the optimal open-loop control sequence dictates an upward decision, whereas the optimal feedback decision is D. Therefore, the expected value E of the open-loop-optimal feedback control scheme is as in Figure 9.2 except that U rather than D is chosen at vertex A. Consequently, at vertex A, we have the result

$$E = \tfrac{1}{4}(0 + 84) + \tfrac{3}{4}(10 + 75) = 84\tfrac{3}{4}.$$

We assert on the basis of this example that

(1) The optimal *open-loop* scheme incorporating no use of subsequent information about actual transitions generally yields a large expected cost, but is the best that can be done without using more information.

(2) The optimal *feedback* scheme where the state is assumed known when the decision is made yields the smallest possible expected cost for this type of stochastic problem since it uses all available information.

(3) The hybrid *open-loop-optimal feedback* scheme yields an intermediate expected cost. Although feedback is used, the fact that feedback is to be used is withheld from the computation determining the control decisions, and this results in a control scheme that is inferior to true optimal feedback control. There is little to recommend this scheme since it uses as much information as the true feedback solution. We presented it to protect the reader from mistakenly believing that this is an alternative way of computing the true optimal feedback solution.

Problem 9.1. Compare the approximate number of calculations using (9.1) and (9.2) for an N-stage stochastic problem to the number required to solve a deterministic path problem on the same network.

Problem 9.2. Try to construct a forward dynamic-programming algorithm for the optimal feedback solution of the problem in Section 2. Check its validity by using it to solve the problem numerically.

Problem 9.3. Consider the general nonnegative data $a_u(x, y)$, $a_d(x, y)$ for a problem of duration N stages on a triangular network of the type we have been treating. Let p_u be the probability the decision U goes up and p_d be the probability

that decision D goes down. Use dynamic programming to determine the feedback policy that maximizes the probability that the cost is less than or equal to Z, for given Z.

6. A Stochastic Stopping-Time Problem

Examination of our ultimate decision problem, that of leading our own personal life, immediately tells us that not only is the outcome of our decisions stochastic, but so also is the duration of the process. Somewhat less morbidly, the life of a company, the duration of a baseball inning, our opportunities to marry are all processes of uncertain duration. To show how such problems are amenable to dynamic-programming solution, we now treat a minimum-expected-cost path problem with uncertain stopping time.

On a network of the type shown in Figure 9.3 we start at vertex A (the point $(0, 0)$) and always move diagonally to the right. While outcomes of decisions could be taken to be stochastic, we assume U and D decisions lead, respectively, to diagonally upward and diagonally downward moves with certainty, incurring the cost $a_u(x, y)$ if upward and $a_d(x, y)$ if downward. What is novel about our problem is that when the process starts we do not know whether it will terminate when we reach line B ($x = x_2$) or when we reach line C ($x = x_3$). We do know, however, that the probability of the former is p_B and of the latter, p_C ($= 1 - p_B$), and we assume further that when we reach the line D ($x = x_1$), and prior to making our decision there, we are told at which line the problem is to terminate.

Were we hired as a consultant at some stage during this process, we would first ask our employer the x and y coordinates of the current vertex. Then, if the stage is between x_1 and x_2, including x_1 and x_2, we would also ask what fact about the terminal line had been imparted at stage x_1. If
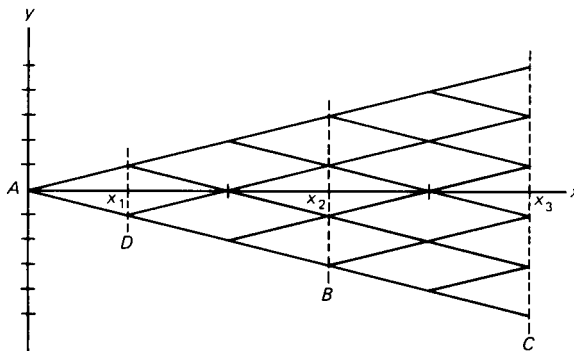
Figure 9.3

$x < x_1$, no information has yet been given, and if $x > x_2$, clearly x_3 is the terminal line. Hence, we have different definitions of the optimal value function, depending on the stage. They are: for $x \leqslant x_1 - 1$,

$F(x, y) =$ the minimum expected cost of the remaining process given that we start at (x, y); $\qquad$ (9.3)

for $x_1 \leqslant x \leqslant x_2$,

$G(x, y, z) =$ the minimum cost of the remaining process given that we start at (x, y) and terminate at z, where $z = x_2$ or $z = x_3$; $\qquad$ (9.4)

and for $x \geqslant x_2$,

$H(x, y) =$ the minimum cost of the remaining process given that we start at (x, y) and terminate at $x = x_3$. $\qquad$ (9.5)

Note that only the first function is an expected cost since the duration is known with certainty once the arguments, including z, are given for the second and third optimal value functions.

The recurrence relation depends on x and is, for $x \leqslant x_1 - 2$,

$$F(x, y) = \min \begin{bmatrix} a_u(x, y) + F(x + 1, y + 1) \\ a_d(x, y) + F(x + 1, y - 1) \end{bmatrix} \qquad (9.6)$$

with the boundary condition

$$F(x_1 - 1, y) = \min \begin{bmatrix} a_u(x_1 - 1, y) + p_B G(x_1, y + 1, x_2) + p_C G(x_1, y + 1, x_3) \\ a_d(x_1 - 1, y) + p_B G(x_1, y - 1, x_2) + p_C G(x_1, y - 1, x_3) \end{bmatrix}$$
$$(9.7)$$

(for the particular case of Figure 9.3 where $x_1 = 1$, Equation (9.6) is never used); for $x_1 \leqslant x \leqslant x_2 - 1$,

$$G(x, y, z) = \min \begin{bmatrix} a_u(x, y) + G(x + 1, y + 1, z) \\ a_d(x, y) + G(x + 1, y - 1, z) \end{bmatrix} \qquad (9.8)$$

with the boundary condition

$$G(x_2, y, x_2) = 0 \quad \text{for all } y, \quad G(x_2, y, x_3) = H(x_2, y); \quad (9.9)$$

and for $x \geqslant x_2$,

$$H(x, y) = \min \begin{bmatrix} a_u(x, y) + H(x + 1, y + 1) \\ a_d(x, y) + H(x + 1, y - 1) \end{bmatrix} \qquad (9.10)$$

with the boundary condition

$$H(x_3, y) = 0 \quad \text{for all } y. \qquad (9.11)$$

We solve by computing backward from x_3, first H, then G (which gives us a different cost and possibly a different decision at each vertex for each of the two terminal situations), and finally F.
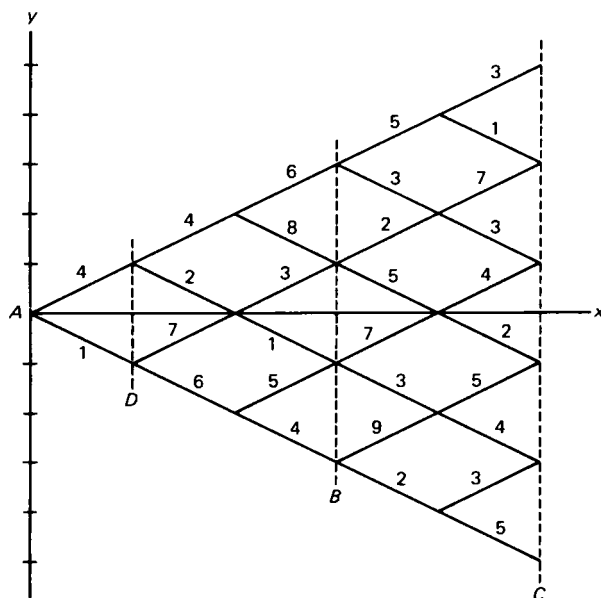
Figure 9.4

Problem 9.4. Solve numerically the problem given in Figure 9.4 with $p_B = \frac{1}{3}$, $p_C = \frac{2}{3}$.

Problem 9.5. Consider a general triangular network of the type shown in Figure 9.4, and suppose that $p_3 =$ the probability termination occurs at $x = x_3$ and $p_4 (= 1 - p_3) =$ the probability termination occurs at $x = x_4$. We find out for sure which is the case just prior to making our decision at either stage x_1 or x_2, with $p_1 =$ the probability we find out at x_1 and $p_2 (= 1 - p_1) =$ the probability we find out at x_2. Assuming that $0 < x_1 < x_2 < x_3 < x_4$, give the dynamic-programming formulation.

Problem 9.6. For the same general triangular network suppose that the process can terminate at any stage x, $x = 1, \ldots, N$, where N is fixed and given. When the process starts at $(0, 0)$ we know that p_z $(z = 1, \ldots, N)$ is the probability that the process will end when $x = z$. For each value of z, we find out if $x = z$ is the terminal stage after we arrive at $x = z$. Give the dynamic-programming formulation of this minimum expected cost problem.

7. Problems with Time-Lag or Delay

For many problems, time passes between the making of a decision and its implementation. In production planning problems, implementation of a decision to increase the production level requires such time-consuming

actions as the hiring and training of labor. In inventory problems, the placing of an order and its arrival are separated by time for transmission and processing of the order and by shipping time, as well as possibly by production time. For deterministic path problems, such delays present no difficulty since if decision d is optimal at stage k in the open-loop representation of the solution and it takes l stages to implement a decision, we would simply displace our decisions l stages and make decision d at stage $k - l$. However, for feedback control of stochastic processes, we shall not know at stage $k - l$ what the state will be at stage k, so we cannot merely displace the decision l stages.

To illustrate the use of dynamic programming on such problems, we now consider the minimum cost stochastic path problem shown in Figure 9.5 where our decision at stage k is implemented at stage $k + 2$, no matter what the state at stage $k + 2$. We are using feedback since our decision at stage k will depend on the vertex at stage k, along with other information. The ideas developed here will be used in Chapter 11 where stochastic inventory problems with delay are considered.

Decision U results in a diagonally upward move when it is implemented two stages later with probability $\frac{3}{4}$ and it results in a downward move two stages later with probability $\frac{1}{4}$. Decision D is like the U decision except the probabilities, $\frac{3}{4}$ and $\frac{1}{4}$, are interchanged. Since the decision at $(0, 0)$ is not implemented until stage 2 (i.e., $x = 2$), we must further specify what happens at stages 0 and 1. We assume that at $(0, 0)$,
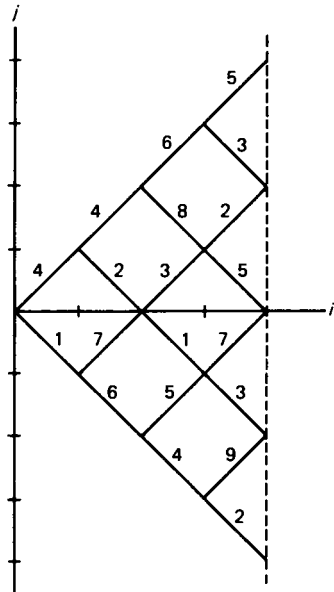

Figure 9.5

(1, 1), and (1, − 1) the probability of moving diagonally upward and the probability of moving diagonally downward each equal $\frac{1}{2}$.

Solution consists of decisions at (0, 0), (1, 1), and (1, − 1) since these decisions are implemented, due to the delay of two stages, at stages 2 and 3. No decisions are called for at stages 2 and 3 since the problem ends before they are implemented.

Were we hired as a consultant while a process of the above type was in progress, we would certainly need to know the current vertex. Furthermore, we would want to know the decision made two stages ago by our employer since that decision will be implemented at the current vertex and will influence the probabilities of the vertices two stages hence, when our first decision will be implemented. Similarly, we would want to know the decision made by our employer at the previous stage since it will be implemented at the next vertex. This leads to the following definition of the optimal expected value function for the general situation where our employer has made the previous two decisions and we are to make decisions henceforth (this situation never occurs in our particular example of only four-stages duration):

$S(i, j, d_1, d_2) =$ the minimum expected cost of the remaining
  process given that we start at (i, j), d_1 was
  the decision made at stage $i − 1$, and d_2 was
  the decision made at stage $i − 2$.

By the principle of optimality, writing each of the four possible sets of previous two decisions separately, we have

$S(i, j, U, U) = \frac{3}{4} a_u(i, j) + \frac{1}{4} a_d(i, j)$
$$+ \min \begin{bmatrix} \frac{3}{4} S(i + 1, j + 1, U, U) + \frac{1}{4} S(i + 1, j − 1, U, U) \\ \frac{3}{4} S(i + 1, j + 1, D, U) + \frac{1}{4} S(i + 1, j − 1, D, U) \end{bmatrix},$$

$S(i, j, U, D) = \frac{1}{4} a_u(i, j) + \frac{3}{4} a_d(i, j)$
$$+ \min \begin{bmatrix} \frac{1}{4} S(i + 1, j + 1, U, U) + \frac{3}{4} S(i + 1, j − 1, U, U) \\ \frac{1}{4} S(i + 1, j + 1, D, U) + \frac{3}{4} S(i + 1, j − 1, D, U) \end{bmatrix},$$

$S(i, j, D, U) = \frac{3}{4} a_u(i, j) + \frac{1}{4} a_d(i, j)$
$$+ \min \begin{bmatrix} \frac{3}{4} S(i + 1, j + 1, U, D) + \frac{1}{4} S(i + 1, j − 1, U, D) \\ \frac{3}{4} S(i + 1, j + 1, D, D) + \frac{1}{4} S(i + 1, j − 1, D, D) \end{bmatrix},$$

$S(i, j, D, D) = \frac{1}{4} a_u(i, j) + \frac{3}{4} a_d(i, j)$
$$+ \min \begin{bmatrix} \frac{1}{4} S(i + 1, j + 1, U, D) + \frac{3}{4} S(i + 1, j − 1, U, D) \\ \frac{1}{4} S(i + 1, j + 1, D, D) + \frac{3}{4} S(i + 1, j − 1, D, D) \end{bmatrix}.$$

(9.12)

Note that the first two terms on the right-hand side, the expected cost of the next transition, depend on the fourth argument of S, the decision made two stages earlier, and that the decision at stage i (U for the top line and D for the bottom line in the bracket following the min) appears as the third argument of S at stage $i + 1$, while the third argument of S at stage i becomes the fourth argument of S at stage $i + 1$.

For processes starting at stage 0 or 1 we have assumed that upward and downward transitions each occur with probability $\frac{1}{2}$, so

$$S(1, j, U, \_) = \tfrac{1}{2} a_u(1, j) + \tfrac{1}{2} a_d(1, j)$$
$$+ \min \left[\begin{array}{c} \tfrac{1}{2} S(2, j + 1, U, U) + \tfrac{1}{2} S(2, j - 1, U, U) \\ \tfrac{1}{2} S(2, j + 1, D, U) + \tfrac{1}{2} S(2, j - 1, D, U) \end{array} \right],$$

$$S(1, j, D, \_) = \tfrac{1}{2} a_u(1, j) + \tfrac{1}{2} a_d(1, j)$$
$$+ \min \left[\begin{array}{c} \tfrac{1}{2} S(2, j + 1, U, D) + \tfrac{1}{2} S(2, j - 1, U, D) \\ \tfrac{1}{2} S(2, j + 1, D, D) + \tfrac{1}{2} S(2, j - 1, D, D) \end{array} \right], \quad (9.13)$$

$$S(0, 0, \_, \_) = \tfrac{1}{2} a_u(0, 0) + \tfrac{1}{2} a_d(0, 0)$$
$$+ \min \left[\begin{array}{c} \tfrac{1}{2} S(1, 1, U, \_) + \tfrac{1}{2} S(1, -1, U, \_) \\ \tfrac{1}{2} S(1, 1, D, \_) + \tfrac{1}{2} S(1, -1, D, \_) \end{array} \right].$$

The boundary condition, assuming the process ends when $i = 4$, is most easily written as

$$S(4, j, d_1, d_2) = 0 \qquad \text{for all } j, d_1, \text{ and } d_2. \tag{9.14}$$

In this case, the value of S will be computed at stage 3 for various decisions at stage 2 where these decisions are irrelevant and not really made, but the answer will be correct. More complicated to write, but easier to use for hand computation, are formulas for S at stage 2 of the form

$$S(2, j, U, U) = \tfrac{9}{16} \left[a_u(2, j) + a_u(3, j + 1) \right] + \tfrac{3}{16} \left[a_u(2, j) + a_d(3, j + 1) \right]$$
$$+ \tfrac{3}{16} \left[a_d(2, j) + a_u(3, j - 1) \right] + \tfrac{1}{16} \left[a_d(2, j) + a_d(3, j - 1) \right]. \tag{9.15}$$

Problem 9.7. Use (9.12), (9.13), and boundary conditions of the type of (9.15) to solve the problem described in the text for the network shown in Figure 9.5.

The reader should now test his understanding of the concepts and methods of this chapter on the following problems.

Problem 9.8. Determine the feedback policy that minimizes the expected cost of going from A to line B in the network in Figure 9.6 where the cost of a path is the sum of its arc numbers plus 1 for each change in direction, and where at each vertex there are two admissible decisions. Decision U goes diagonally up with
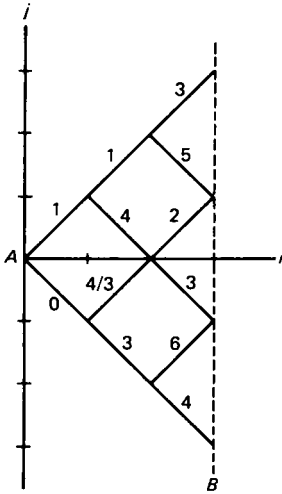
Figure 9.6

probability $\frac{2}{3}$ and down with probability $\frac{1}{3}$ and decision D goes up with probability $\frac{1}{3}$ and down with probability $\frac{2}{3}$.

Problem 9.9. On the network shown in Figure 9.7, you are to find a decision at each node that minimizes the expected cost of the resulting path from A to line B. At each node you have two possible decisions. Decision U taken at a node has probability $\frac{1}{2}$ of resulting in a transition to the node diagonally upward and, if so, a cost equal to the number on the included arc is incurred; and it has probability $\frac{1}{2}$ of resulting in a transition leaving the state unchanged (i.e., staying at the node in question for at least one more stage), in which case a cost of 1 is assessed and the process continues. Decision D has probability $\frac{1}{3}$ of a diagonally downward transition with cost as indicated on the downward arc; and it has probability $\frac{2}{3}$ of staying at the same node, with a cost of 1 assessed, and the process continues.
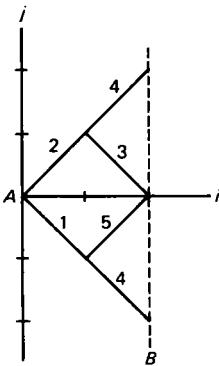
Figure 9.7
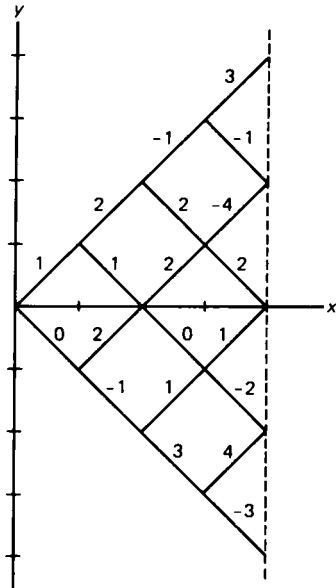
Figure 9.8

Problem 9.10. Consider the network shown in Figure 9.8. As in Section 2, decision U at (x, y) results in a move to $(x + 1, y + 1)$ at a cost $a_u(x, y)$ with probability $\frac{3}{4}$ and it results in a move to $(x + 1, y - 1)$ at a cost $a_d(x, y)$ with probability $\frac{1}{4}$. Decision D differs from U only in that the $\frac{3}{4}$ and $\frac{1}{4}$ are interchanged. A third decision, decision S, stops the process at the vertex at which the decision is made and no further cost is incurred. Find the feedback policy that minimizes the expected cost.

Problem 9.11. Solve the problem in Figure 9.1 if, after arriving at each vertex and before choosing decision U or D at that vertex, you are told with certainty in which direction you will move if you choose decision U and in which direction you will move if you choose D. The certain result of decision U at a vertex is determined randomly after you arrive there, with probability $\frac{3}{4}$ that decision U will mean "go diagonally up" and with probability $\frac{1}{4}$ that U will mean "go diagonally down." Similarly and independently, the certain result of decision D at the vertex is randomly determined, with probability $\frac{1}{4}$ that decision D will mean "go diagonally up" and with probability $\frac{3}{4}$ that it will mean "go diagonally down." Hence, it could turn out at some vertices that both U and D mean the same thing, in which case you really have no choice.

The importance of models in which the random event at each stage precedes the decision will become clear in Chapter 10, Section 3.

Chapter 10

STOCHASTIC EQUIPMENT INSPECTION
AND REPLACEMENT MODELS

1. Introduction

In this chapter, and its problems, we illustrate the practicality and flexibility of stochastic dynamic programming. In the text we treat stochastic equipment inspection and replacement models. In the problems we introduce further applications, not only to problems of business but also to games of pleasure. Stochastic games are more interesting and rewarding than deterministic ones since optimal play will not *always* achieve the most desirable possible outcome, thereby making it easier to exploit opponents *in the long run*.

2. Stochastic Equipment-Replacement Models

We begin with a simple model of the type of Chapter 2, Section 1, where we either keep or replace our current machine at the start of each year i, $i = 1, \ldots, N$. We assume that the operating cost is a random variable, dependent upon the age of the machine. We further assume that our machine may suffer a catastrophic failure at the end of any year, and then it must be replaced by a new machine. The data defining our problem are

N = the duration of the process,

y = the age of the machine, in working order, with which we start year 1,

$n(i, j)$ = the probability that the net operating cost during the year is j, $j = 0, 1, \ldots, J$, given that the machine is of age i at the start of the year,

p = the purchase price of a new machine,

$t(i)$ = the trade-in value of a machine, in working order, just turned age i,

$u(i)$ = the trade-in value of a machine, in failed condition, just turned age i,

$q(i)$ = the probability that a machine, in working order, of age i at the start of a year fails at the end of the year,

$s(i)$ = the salvage value at the start of year $N + 1$ of a working machine just turned age i,

$v(i)$ = the salvage value at the start of year $N + 1$ of a failed machine just turned age i.

To formulate the problem of minimizing the expected cost for the above situation, we define

$S(i, k)$ = the minimum expected cost of the remaining process if we start year k with a machine, in working order, of age i. $\qquad$ (10.1)

Then, by the principle of optimality, for $k = 1, \ldots, N - 1$; $i = 1, \ldots, k - 1$ and $i = y + k - 1$:

$$S(i, k) = \min \begin{bmatrix} B: \ p - t(i) + \displaystyle\sum_{j=0}^{J} jn(0, j) + q(0)\{ p - u(1) + S(0, k + 1)\} \\[2mm] + \{1 - q(0)\}S(1, k + 1) \\[4mm] K: \ \displaystyle\sum_{j=0}^{J} jn(i, j) + q(i)\{ p - u(i + 1) + S(0, k + 1)\} \\[2mm] + \{1 - q(i)\}S(i + 1, k + 1) \end{bmatrix};$$

(10.2)

for $i = 0$:

$$S(0, k) = \sum_{j=0}^{J} jn(0, j) + q(0)\{ p - u(1) + S(0, k + 1)\}$$
$$+ \{1 - q(0)\}S(1, k + 1);$$
(10.3)

and the boundary condition is

$$S(i, N) = \min \begin{bmatrix} B: \ p - t(i) + \displaystyle\sum_{j=0}^{J} jn(0, j) - q(0)v(1) \\[2mm] - \{1 - q(0)\}s(1) \\[4mm] K: \ \displaystyle\sum_{j=0}^{J} jn(i, j) - q(i)v(i + 1) \\[2mm] - \{1 - q(i)\}s(i + 1) \end{bmatrix}.$$

(10.4)

If the expected operating cost of an i-year-old machine $\sum_{j=0}^{J} jn(i, j)$ is computed for all i before the dynamic-programming solution begins, use of (10.2)–(10.4) entails only about twice the computation of the analogous deterministic model of Chapter 2.

Let us elaborate the above model. We assume that, in case of catastrophic failure, we can either replace the machine, as above, or we can pay an amount $w(i)$, if the machine has just turned age i, to restore the machine to working order. Defining S as in (10.1), we have, for $k = 1, \ldots, N - 1$; $i = 1, \ldots, k - 1$ and $i = y + k - 1$:

$$S(i, k) = \min \begin{bmatrix} B\colon \ p - t(i) + \sum_{j=0}^{J} jn(0, j) \\ \qquad + q(0) \min \left\{ \begin{array}{c} p - u(1) + S(0, k + 1) \\ w(1) + S(1, k + 1) \end{array} \right\} \\ \qquad + \{1 - q(0)\} S(1, k + 1) \\ K\colon \ \sum_{j=0}^{J} jn(i, j) + q(i) \min \left\{ \begin{array}{c} p - u(i + 1) + S(0, k + 1) \\ w(i + 1) + S(i + 1, k + 1) \end{array} \right\} \\ \qquad + \{1 - q(i)\} S(i + 1, k + 1) \end{bmatrix};$$

$$(10.5)$$

for $i = 0$:

$$S(0, k) = \sum_{j=0}^{J} jn(0, j) + q(0) \min \left\{ \begin{array}{c} p - u(1) + S(0, k + 1) \\ w(1) + S(1, k + 1) \end{array} \right\}$$
$$+ \{1 - q(0)\} S(1, k + 1). \qquad (10.6)$$

Boundary condition (10.4) is unaffected by the elaboration. The optimal policy is now not only a "buy" or "keep" decision for each state and stage, but also an accompanying "buy" or "restore" decision to be used in case of catastrophic failure. The computation required by (10.5) and (10.6) is only about 10% greater than that required by (10.2) and (10.3). Since the decision concerning the random event "catastrophic failure" is made *after* observing the event, the terms involving $q(i)$ in (10.5) and (10.6) calculate the expected value of the minimum rather than the standard minimum of the expected value. See Problem 9.11 where we first introduced a situation where the random event at a stage preceded the decision.

Problem 10.1. Suppose, in the above model, that if a catastrophic failure occurs when the machine turns age i the probability is $m(i, l)$, $l = 1, \ldots, L$, that the cost of restoring the machine to working order is l. We can either find out the actual cost once failure occurs by paying an inspection cost c and then, based on

the actual cost, we can decide whether to buy or restore, or we can immediately buy once catastrophic failure occurs. Hence, we can pay, if we choose, to have the random event precede the decision. Give the dynamic-programming formulation.

3. An Inspection and Replacement Problem

We consider first a problem involving inspection and replacement of unreliable components of a system. Sometimes, when inspecting or replacing one component of a system, money or time can be saved by simultaneously doing the same for other components. We again encounter models where some random events follow the decision at each stage, but others precede the decision.

Suppose that an automatic sprinkling system, to be used in case of fire, consists of two unreliable parts, numbered 1 and 2. If either part has failed, the system will fail to work if called upon. Only by examination of a part at the beginning of a time period can its condition, good or failed, be determined. If a part fails, it does so at the end of a period. A part just examined and found good, is as good as a new part, and in either case such a part is said to be of age 0. A part not examined for j periods is said to be of age j. The probability of part i, $i = 1$ or 2, being good at the beginning of a period, given that it was good at the beginning of the previous period, is p_i.

It takes c_i periods to examine part i for failure (examination always commences at the beginning of a period) and c_{12} periods ($c_{12} < c_1 + c_2$) to examine both parts simultaneously. It takes r_i periods to replace part i and r_{12} periods ($r_{12} < r_1 + r_2$) to replace both simultaneously. (All c's and r's are integral multiples of the basic time period.) Inspection and replacement cannot be done simultaneously. While inspection or replacement of one or both parts is taking place the system is not working. No part ages while the system is not working due to inspection or replacement. Parts can be replaced, if desired, without inspection.

Initially part 1 is of age j_0 and part 2 is of age k_0, and we wish to find an inspection and replacement policy that maximizes the expected number of periods during the next N periods that the system is working.

To solve this problem using dynamic programming, we define the optimal expected value function by

$S_i(j, k) =$ the maximum expected number of periods from period i through period N during which the system is working, given that the system starts period i with part 1 last known to be in good order j working periods previously and with part 2 last known to be in good order k working periods previously.

It is convenient to define several other functions:

$F_i(j)$ = the maximum expected number of remaining periods during which the system is working, given that the system starts period i with part 1 last known to be in good order j working periods previously and part 2 just inspected and in failed condition,

$G_i(k)$ = the same as F except 1 has failed and 2 is of age k,

H_i = the same as F except both parts have failed.

We define the following possible decisions: O means "let the system operate as is," $I1$ means "inspect part 1," $I2$ means "inspect part 2," $I12$ means "inspect both parts," $R1$ means "replace part 1," $R2$ means "replace part 2," and $R12$ means "replace both parts."

We can now write for $j > 0$, $k > 0$:

$$S_i(j, k) = \max \begin{bmatrix} O\text{:} & p_1^j p_2^k + S_{i+1}(j+1, k+1) \\ I1\text{:} & p_1^j S_{i+c_1}(0, k) + (1 - p_1^j)G_{i+c_1}(k) \\ I2\text{:} & p_2^k S_{i+c_2}(j, 0) + (1 - p_2^k)F_{i+c_2}(j) \\ I12\text{:} & p_1^j p_2^k S_{i+c_{12}}(0, 0) + (1 - p_1^j)p_2^k G_{i+c_{12}}(k) \\ & \quad + p_1^j(1 - p_2^k)F_{i+c_{12}}(j) + (1 - p_1^j)(1 - p_2^k)H_{i+c_{12}} \\ R1\text{:} & S_{i+r_1}(0, k) \\ R2\text{:} & S_{i+r_2}(j, 0) \\ R12\text{:} & S_{i+r_{12}}(0, 0) \end{bmatrix},$$

$$F_i(j) = \max \begin{bmatrix} I1\text{:} & p_1^j F_{i+c_1}(0) + (1 - p_1^j)S_{i+c_1+r_{12}}(0, 0) \\ R2\text{:} & S_{i+r_2}(j, 0) \\ R12\text{:} & S_{i+r_{12}}(0, 0) \end{bmatrix}, \qquad (10.7)$$

$$G_i(k) = \max \begin{bmatrix} I2\text{:} & p_2^k G_{i+c_2}(0) + (1 - p_2^k)S_{i+c_2+r_{12}}(0, 0) \\ R1\text{:} & S_{i+r_1}(0, k) \\ R12\text{:} & S_{i+r_{12}}(0, 0) \end{bmatrix},$$

$$H_i = S_{i+r_{12}}(0, 0).$$

When one or more argument is 0, some options are of no interest, so

$$S_i(0, k) = \max \begin{bmatrix} O\text{:} & p_2^k + S_{i+1}(1, k+1) \\ I2\text{:} & p_2^k S_{i+c_2}(0, 0) + (1 - p_2^k)F_{i+c_2}(0) \\ R2\text{:} & S_{i+r_2}(0, 0) \end{bmatrix},$$

$$S_i(j, 0) = \max \begin{bmatrix} O: & p_1^j + S_{i+1}(j+1, 1) \\ I1: & p_1^j S_{i+c_1}(0, 0) + (1 - p_1^j) G_{i+c_1}(0) \\ R1: & S_{i+r_1}(0, 0) \end{bmatrix},$$

$$S_i(0, 0) = 1 + S_{i+1}(1, 1),$$
$$F_i(0) = S_{i+r_2}(0, 0),$$
$$G_i(0) = S_{i+r_1}(0, 0).$$

(10.8)

Note that if the system is operated, the probability p of its working equals the expected good time during the period since the good time is 1 with probability p and 0 with probability $1 - p$. The boundary condition is

$$S_i(j, k) = F_i(j) = G_i(k) = H_i = 0$$
$$\text{for all} \quad j, k \quad \text{and for } i = N + 1, N + 2, \ldots . \quad (10.9)$$

The formulas for F, G, and H can be substituted into the recurrence relation for S, giving one long, complicated formula. Defining additional functions clarifies the reasoning behind the formula. This model becomes computationally prohibitive for systems containing more than about three unreliable parts.

Problem 10.2. You are to produce N good bottles using a machine that contains two molds and that simultaneously produces one bottle from each mold. Let p_i be the probability of *successfully* producing a bottle using a mold that has already produced i bottles, $p_0 = 1$. If a mold fails in service, a faulty bottle is produced and a cost c_2 is incurred. Further, when a mold fails, the machine must be stripped down for replacement, a process that costs c_3 for the labor involved and c_4 for the lost production time. A replacement mold costs c_1. Once the machine is stripped, the cost of replacing the second mold is the cost of the mold alone.

By replacing molds before failure, the faulty bottle expense c_2 can be avoided. However such a policy involves more new molds and requires more labor than replacing only after failure.

Write a dynamic-programming recurrence relation and boundary conditions that yield the policy that minimizes the expected cost of producing N good bottles starting with two new molds. If you end up with $N + 1$ bottles, the last one has neither cost nor value.

Problem 10.3. You are about to dispatch a transport plane to an overseas base with a cargo that is to consist of replacement parts for airplanes. There are N types of replacement parts. A penalty c_i is incurred if a part of type i is needed at the base and not available. The demand for the ith part is a random variable Z_i, where $p_i(z)$ is the probability that $Z_i = z$ for $z = 0, 1, \ldots, d_i$. Each item of type i weighs w_i and the transport plane has capacity W.

Give a dynamic-programming formulation of the problem of loading the transport so as to minimize the expected cost due to shortages at the base. (Does

the most efficient algorithm involve a sequence of functions of one state variable or just one function of one variable, computed recursively?)

Problem 10.4. You are given a budget of X dollars to allocate to N winner-take-all Presidential primaries. If x_i dollars are allocated to primary i, with probability $p_i(x_i)$ the candidate wins and gets v_i delegates. With probability $1 - p_i(x_i)$ the candidate loses and gets no delegates. First primary 1 occurs and is either won or lost, then primary 2, etc. Use dynamic programming to maximize the probability that the candidate's delegate total is greater than or equal to V, a given number less than $\sum_{i=1}^{N} v_i$.

Problem 10.5. Suppose you have \$3 and are allowed to make a sequence of four gambles. At each gamble you can bet \$0, \$1, etc., in increments of \$1 up to and including the amount you have. When you bet, you get back twice what you bet with probability .6, and lose the amount bet with probability .4. Write a recurrence relation that can be used to determine what betting strategy maximizes the probability of finishing with at least \$5. Use it to compute the optimal strategy and maximal probability.

We now give a deterministic problem involving a competitive game, to illustrate the use of dynamic programming in sequential situations where the players alternate moves. Then we give a more interesting stochastic version of the game.

Problem 10.6. N matchsticks are placed in a table. Michael moves first and picks up either 1, 2, or 3 matchsticks. Then Elise picks up either 1, 2, or 3 of the remaining matchsticks. Then Michael picks up either 1, 2, or 3 of the remaining matchsticks, then Elise, etc. The object is to force your opponent to take the last matchstick. Give a dynamic-programming solution procedure, assuming neither player ever makes a mistake. Hint: Let S_i denote the value of the game to the player whose move it is when i matchsticks remain and let the value of a winning position be 1 and a losing position be 0. Clearly, $S_1 = 0$.

Problem 10.7. The game is played as in Problem 10.6 except that after each player's move, if any matchsticks are left, a fair coin is flipped and if it is heads the player who just moved must remove one more matchstick.

You should now be able to invent and solve numerous variations on this game. If you play optimally and your opponent does not, you should let the number of matchsticks and/or the choice of who moves first be randomly determined. You will still win in the long run and your opponent will not quit quite so soon.

Problem 10.8. Louise wishes to minimize her expected score in the following game. Nine markers, numbered 1 through 9, are placed on a table. Louise rolls a pair of six-sided dice. If the numbers on the dice sum to k ($k = 2, \ldots, 12$), she removes from the table any set of markers whose numbers also sum to k. (If she

rolls a 4 and a 5, she may remove the 9 marker, or she may remove the 2, 3, and 4 markers, or any of several other possibilities.) She then rolls the dice again and removes markers as above, continuing until no set of remaining markers adds to the sum of the dice. The game then ends and her score is the sum of the numbers on the remaining markers. Give a dynamic-programming procedure for determining Louise's optimal strategy.

Problem 10.9. Suppose that Louise has played the game of Problem 10.8 and her score is z. Now Will places the nine markers on the table and plays by the same rules. However, his object is to obtain a score smaller than Louise's. Use dynamic programming to find the strategy that maximizes Will's probability of beating Louise, given that her score is z.

Problem 10.10. A woman wishes to take one of the N men she knows to a party. If man i is the jth man invited, she feels that he will accept the invitation with probability p_{ij}. (All N men know whenever any one is asked.) If man i accepts, the process ends and the value to the woman is v_i. If he declines, another man is asked (no one is asked more than once) until either one accepts or all N available men have declined (value 0). Use dynamic programming to determine the order of invitations that maximizes her expected value.

Try your procedure on the case $N = 2$; $v_1 = 10$, $p_{11} = \frac{3}{5}$, $p_{12} = \frac{1}{2}$; $v_2 = 8$, $p_{21} = \frac{1}{2}$, $p_{22} = 0$. Note that the woman values man 1 more highly than 2 and that he is more likely to accept either a first or a second invitation, and that man 2 is more proud, refusing any second invitation. What is the moral of the optimal solution?

Chapter 11

DYNAMIC INVENTORY SYSTEMS

1. The Nature of Inventory Systems

In this and the next chapter we analyze multistage inventory systems by means of dynamic programming. Consider a company that sells a single product and would like to decide how many items to have in inventory for each of the next n time periods. The length of a time period may be a week, a month, etc., and the number of periods for which the company would like to schedule its inventory (in this case n) is called the *planning horizon*.

The demand for the product in period i, D_i, is a nonnegative integer-valued random variable with $p_i(d)$ denoting the probability that $D_i = d$, and it is assumed that $D_1, D_2, \ldots, D_n$ are independent. At the beginning of each period the company reviews the inventory level and decides how many items to order from its supplier. (An alternative assumption would be that the company manufactures its own items. Thus, at the beginning of each period it must decide how many items to manufacture.) Let x_i be the amount of inventory on hand and on order before ordering in period i. If an order is placed for z_i (a nonnegative integer) items, it arrives in period $i + \lambda$ where λ, a known nonnegative integer, is the *delivery lag*. It is assumed that $\lambda < n$ and that the company never orders in periods $n - \lambda + 1, n - \lambda + 2, \ldots, n$. Let y_i be the amount of inventory on hand and on order after ordering in period i. Thus, $y_i = x_i + z_i$. In some cases there may be restrictions on the amount ordered z_i or the inventory level y_i. Let w_i be the amount of inventory on hand (not including what is still on order) in period i after the order from period $i - \lambda$, $z_{i-\lambda}$, has been delivered but before the demand occurs. Thus, w_i is the amount of inventory actually available to meet demand in period i. If $\lambda = 0$, then $w_i = y_i$; and if $\lambda > 0$, then $w_{i+\lambda} = y_i - (d_i + d_{i+1} + \cdots +$

Period i

Inventory on hand

Inventory on hand and on order
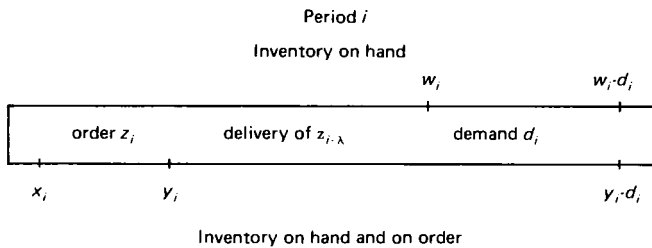
Figure 11.1

$d_{i+\lambda-1}$), where d_i is the actual demand in period i. The sequence of events (during a time period) for $\lambda \geqslant 0$ is shown in Figure 11.1.

To determine x_{i+1}, it is necessary to make some assumption about what happens when $d_i > w_i$. We shall consider two different cases. If all demand unfilled at the end of period i is eventually satisfied by future deliveries, then this is called the *backlogged* case. In this case, $x_{i+1} = y_i - d_i$ and may be negative. If all demand unfilled at the end of period i is assumed lost, then this is called the *lost sales* case. In this case, if $\lambda = 0$, then $x_{i+1} = \max(y_i - d_i, 0)$; and if $\lambda > 0$, then, as we shall see in Section 3, the situation is more complex. (In the lost sales case it is assumed that unfilled demand in period i is lost after $w_i - d_i$ and $y_i - d_i$ have been determined but before the beginning of period $i + 1$. Also, the expression for $w_{i+\lambda}$ that was given above is not valid.)

There are three costs associated with operating the above inventory system. If z items are ordered in period i, then there is an ordering cost $c_i(z)$ which is incurred at the time of delivery in period $i + \lambda$. (It is also possible to assume that the ordering cost is incurred at the time of ordering.) If the net inventory on hand at the end of period i, namely $w_i - d_i$, is nonnegative, then there is a holding cost $h_i(w_i - d_i)$. The holding cost includes such costs as warehouse rental, insurance, taxes, and maintenance. It also includes the cost of having capital tied up in inventory rather than having it invested elsewhere. If $w_i - d_i$ is negative (there is unfilled demand at the end of period i), then there is a shortage (or penalty) cost $\Pi_i(d_i - w_i)$ and a minimal holding cost $h_i(0)$. It is assumed that $\Pi_i(0) = 0$. The shortage cost includes the cost due to extra record keeping in the backlogged case and the cost due to loss of revenue in the lost sales case. For some inventory systems it may also include a cost due to the loss of customers' goodwill.

Up to this point in the book, in order not to obscure other fundamental concepts, we have always assumed that a cost c incurred in period j is equivalent to a cost c incurred in any future period. However, if the length of a period is sufficiently long and/or the interest rate on

invested capital is sufficiently large, then this may not be a good assumption. Let i be the interest rate per time period on invested capital. That is, 1 dollar invested at the beginning of period j is worth $1 + i$ dollars at the beginning of period $j + 1$. Let $\alpha = 1/(1 + i)$. The quantity α is called the *discount factor*, and it follows that $0 < \alpha \leq 1$. (We have, in effect, been assuming that $\alpha = 1$ up to this point in the book.) Thus, a cost c incurred in period $j + 1$ has a discounted value of αc in period j. Similarly, a cost c incurred in period $j + k$ has a discounted value of $\alpha^k c$ in period j. This is equivalent to the statement that an investment of $\alpha^k c$ dollars in period j at interest rate i will yield c dollars k periods later ($c = \alpha^k c(1 + i)^k$).

An ordering policy is a rule for deciding how much to order at the beginning of each time period. Corresponding to each ordering policy is an expected total discounted cost for the n time periods. The objective of the company is to choose an *optimal* ordering policy, i.e., one that minimizes the expected total discounted cost.

In the remainder of this chapter we shall see how dynamic programming can be used to obtain an optimal ordering policy for the above inventory systems. We shall also consider an inventory model for which the delivery lag is a random variable. In Chapter 12 we shall see that by making special assumptions about the cost functions, it is possible to characterize the *form* of an optimal ordering policy. In particular, we shall see that under certain circumstances an optimal ordering policy is specified by only one or two numbers per period. Knowing the form of the optimal policy will in turn lead to a reduced number of calculations.

Before concluding this section, there is one additional matter to discuss. Given that $p_i(d)$ is the probability that $D_i = d$ ($i = 1, 2, \ldots, n$) and, furthermore, that the D_i are independent random variables, then the probability that

$$D_i + D_{i+1} + \cdots + D_j = d \qquad (i = 1, 2, \ldots, n; \quad j = i, i + 1, \ldots, n)$$

is called the *convolution* of $p_i(d), p_{i+1}(d), \ldots, p_j(d)$ and is denoted by $p_{i,j}(d)$. We can recursively compute $p_{i,j}(d)$ (for $d = 0, 1, \ldots$) as follows:

$$p_{i,i}(d) \equiv p_i(d),$$

$$p_{i,k}(d) = \sum_{l=0}^{d} p_{i,k-1}(l)p_k(d - l) \qquad (k = i + 1, i + 2, \ldots, j).$$

We shall use $p_{i,j}(d)$ in Sections 3 and 4 and also in Chapter 12.

2. Models with Zero Delivery Lag

In this section we shall see how dynamic programming can be used to determine an optimal ordering policy when the delivery lag λ is zero or

negligible relative to the length of a time period. (The more realistic assumption of $\lambda \geqslant 1$ will be considered in the next section.)

Let us begin with the backlogged case. We shall assume that if there is a nonnegative inventory of v items at the end of period n (the beginning of period $n + 1$), then the v items may be returned and the company receives a salvage value of $s(v)$ (incurs a cost of $-s(v)$). Conversely, if there is a backlog of v items at the end of period n, then the backlog can be satisfied by purchasing v items at a cost of $b(v)$. If $t(v)$ is the terminal cost when there is an inventory of v ($v = \ldots, -2, -1, 0, 1, 2, \ldots$) items at the end of period n, then $t(v)$ is given by

$$t(v) = \left[\begin{array}{ll} -s(v) & \text{if} \quad v \geqslant 0, \\ b(-v) & \text{if} \quad v < 0. \end{array} \right.$$

Suppose that at the beginning of period i, x items are on hand (no items are on order since $\lambda = 0$). If we order $y - x$ items in period i, then an ordering cost $c_i(y - x)$ is incurred and y items are available to meet demand. Let $L_i^0(w, d)$ be the holding and shortage cost in period i when w ($w = y$ since $\lambda = 0$) items are on hand and there is a demand d. Similarly, let $L_i^0(w)$ be the expected holding and shortage cost in period i when there are w items on hand to meet demand. Then $L_i^0(w, d)$ and $L_i^0(w)$ are given by

$$L_i^0(w, d) = \left[\begin{array}{ll} h_i(w - d) & \text{if} \quad w - d \geqslant 0, \\ \Pi_i(d - w) + h_i(0) & \text{if} \quad w - d \leqslant 0; \end{array} \right.$$

$$L_i^0(w) = \sum_{d=0}^{\infty} L_i^0(w, d)p_i(d). \tag{11.1}$$

Let us now consider the dynamic-programming formulation for the backlogged case. Suppose that future costs are discounted at a rate of α per period and that the company would like to determine an ordering policy that minimizes the expected total discounted cost for the n-period planning horizon. Define the optimal expected value function $f_i(x)$ as follows:

$f_i(x) =$ the expected total discounted cost (discounted to period i) for periods i through n (plus the terminal cost) if the amount of inventory on hand at the beginning of period i is x (no items are on order since $\lambda = 0$) and an optimal ordering policy is followed. (11.2)

If $D_i = d$, then $x_{i+1} = y - d$ and it is easy to see that $f_i(x)$ satisfies the recurrence relation

$$f_i(x) = \min_{y \geqslant x} \left[c_i(y - x) + L_i^0(y) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y - d)p_i(d) \right]$$

$$(i = 1, 2, \ldots, n). \tag{11.3}$$

For each value of i, $f_i(x)$ is computed for each possible value of x. If $y_i(x) - x$ is the optimal amount to order when $x_i = x$, then $y_i(x)$ is equal to the value of y that minimizes the r.h.s. of (11.3). The appropriate boundary condition for the backlogged case is

$$f_{n+1}(x) = t(x); \tag{11.4}$$

and if x_1 is the specified initial inventory, then $f_1(x_1)$ is the answer to the n-period problem.

Let us consider a numerical example. Suppose that a company would like to schedule its inventory for the next three time periods and that the following data are given:

$p(0) = \frac{1}{4}, \quad p(1) = \frac{1}{2}, \quad p(2) = \frac{1}{4};$

$c(0) = 0, \quad c(1) = 3, \quad c(2) = 5, \quad c(3) = 6, \quad c(z) = \infty \quad \text{for} \quad z \geqslant 4;$

$h(v) = v \quad \text{for} \quad v \geqslant 0; \quad \Pi(v) = 5v \quad \text{for} \quad v \geqslant 0;$

$s(v) = 2v \quad \text{for} \quad v \geqslant 0; \quad b(v) = 4v \quad \text{for} \quad v \geqslant 0.$

Notice that all cost functions and the demand distribution are independent of time. Let us also assume that $x_1 = 0$ and $\alpha = 1$. The dynamic-programming solution for this problem is as follows:

$$f_4(x) = -2x \quad \text{for} \quad x = 0, 1, \ldots, 9$$

(the maximum possible inventory is 9);

$$f_4(x) = -4x \quad \text{for} \quad x = -1, -2, \ldots, -6$$

(the maximum possible backlog is 6).

$$f_3(6) = \min_{6 \leqslant y \leqslant 9} \left[c(y-6) + L^0(y) + \sum_{d=0}^{2} f_4(y-d)p(d) \right]$$

$$= \min[0 + 5 - 10, 3 + 6 - 12, 5 + 7 - 14, 6 + 8 - 16]$$

$$= \min[-5, -3, -2, -2] = -5, \qquad y_3(6) = 6;$$

(Actually, it is obvious that you never order when x exceeds the largest demand for the remaining process.)

$$f_3(5) = -4, \qquad y_3(5) = 5;$$
$$f_3(4) = -3, \qquad y_3(4) = 4;$$
$$f_3(3) = -2, \qquad y_3(3) = 3;$$
$$f_3(2) = -1, \qquad y_3(2) = 2;$$
$$f_3(1) = 2, \qquad y_3(1) = 1 \quad \text{or} \quad 2;$$
$$f_3(0) = 4, \qquad y_3(0) = 2 \quad \text{or} \quad 3;$$
$$f_3(-1) = 5, \qquad y_3(-1) = 2;$$
$$f_3(-2) = 8, \qquad y_3(-2) = 1;$$

$$f_3(-3) = 15, \qquad y_3(-3) = 0;$$
$$f_3(-4) = 24, \qquad y_3(-4) = -1.$$

$$f_2(3) = \min_{3 < y \leqslant 6} \left[c(y-3) + L^0(y) + \sum_{d=0}^{2} f_3(y-d)p(d) \right]$$
$$= \min\left[0 + 2 - \tfrac{1}{2}, 3 + 3 - 2, 5 + 4 - 3, 6 + 5 - 4 \right]$$
$$= \min\left[1\tfrac{1}{2}, 4, 6, 7 \right] = 1\tfrac{1}{2}, \quad y_2(3) = 3;$$

$$f_2(2) = 2\tfrac{3}{4}, \qquad y_2(2) = 2;$$
$$f_2(1) = 5\tfrac{1}{4}, \qquad y_2(1) = 1;$$
$$f_2(0) = 7\tfrac{1}{2}, \qquad y_2(0) = 3;$$
$$f_2(-1) = 8\tfrac{3}{4}, \qquad y_2(-1) = 2;$$
$$f_2(-2) = 11\tfrac{1}{4}, \qquad y_2(-2) = 1.$$

Finally,

$$f_1(0) = \min_{0 < y \leqslant 3} \left[c(y) + L^0(y) + \sum_{d=0}^{2} f_2(y-d)p(d) \right]$$
$$= \min\left[0 + 5 + 9\tfrac{1}{16}, 3 + 1\tfrac{1}{2} + 7\tfrac{1}{4}, 5 + 1 + 5\tfrac{3}{16}, 6 + 2 + 3\tfrac{1}{16} \right]$$
$$= \min\left[14\tfrac{1}{16}, 11\tfrac{3}{4}, 11\tfrac{3}{16}, 11\tfrac{1}{16} \right] = 11\tfrac{1}{16}, \qquad y_1(0) = 3.$$

Therefore, the minimum expected (discounted) cost is $11\tfrac{1}{16}$ and it is optimal to order three items at the beginning of period 1. What is optimal to order at the beginning of period 2 depends, of course, on the demand during period 1.

Let us now consider the lost sales case. If there is a nonnegative inventory of v items at the end of period n, then there is still a salvage value of $s(v)$. However, if there is a backlog at the end of period n, no cost is incurred at the beginning of period $n+1$ since excess demand is lost.

Let $f_i(x)$ be as defined by (11.2). If $D_i = d$, then $x_{i+1} = \max(y - d, 0)$ and $f_i(x)$ satisfies the recurrence relation

$$f_i(x) = \min_{y > x} \left[c_i(y-x) + L_i^0(y) + \alpha \sum_{d=0}^{\infty} f_{i+1}(\max(y-d, 0))p_i(d) \right]$$
$$(i = 1, 2, \ldots, n). \tag{11.5}$$

For each value of i, $f_i(x)$ is now computed only for nonnegative values of x. Thus, for $\lambda = 0$ the lost sales case requires fewer computations than the backlogged case. The appropriate boundary condition for the lost sales case is given by

$$f_{n+1}(x) = -s(x) \qquad \text{for} \quad x \geqslant 0. \tag{11.6}$$

3. Models with Positive Delivery Lag

In the previous section it was shown how to find the optimal ordering policy for the case $\lambda = 0$. However, by modifying the formulation slightly, it can also be used when $\lambda = 1$.

Problem 11.1.   Let $\lambda = 1$. Give the dynamic-programming formulation for the backlogged case. (The lost sales case is similar.)

Therefore, let us now assume that $\lambda \geqslant 2$. An order for z items placed at the beginning of period i will arrive in period $i + \lambda$, and it is assumed that the ordering cost is actually incurred at the time of delivery. It is also assumed that the company never orders in periods $n - \lambda + 1$, $n - \lambda + 2$, $\ldots$, n and that any costs incurred at the beginning of period $n + 1$ are given by (11.4) or (11.6) depending on whether one is considering the backlogged case or the lost sales case.

We shall begin by giving two solutions for the backlogged case. The first of these is the standard dynamic-programming formulation (see Chapter 9, Section 7) using one stage variable and λ state variables. Define the optimal expected value function $f_i(w, z_{i-\lambda+1}, \ldots, z_{i-1})$ as follows:

$$f_i(w, z_{i-\lambda+1}, \ldots, z_{i-1}) = \text{the expected total discounted cost}$$

(discounted to period i) for periods i through n (other than the cost due to the delivery of previous orders) given that w items will be on hand in period i after $z_{i-\lambda}$ has been delivered, z_j items were ordered in period j $(j = i - \lambda + 1, i - \lambda + 2, \ldots, i - 1)$, and an optimal ordering policy is followed.   (11.7)

If $L_i^0(w)$ is as given by (11.1), then it is easy to see that f_i satisfies the recurrence relation

$$f_i(w, z_{i-\lambda+1}, \ldots, z_{i-1})$$

$$= \min_{z > 0} \left[\alpha^\lambda c_i(z) + L_i^0(w) \right.$$

$$\left. + \alpha \sum_{d=0}^{\infty} f_{i+1}(w - d + z_{i-\lambda+1}, z_{i-\lambda+2}, \ldots, z_{i-1}, z) p_i(d) \right]$$

$$(i = 1, 2, \ldots, n - \lambda), \tag{11.8}$$

where z, of course, is the amount ordered in period i. The boundary condition, which must now include the expected holding and shortage

costs in periods $n - \lambda + 1$, $n - \lambda + 2$, ..., n as well as the terminal cost, is given by

$$f_{n-\lambda+1}(w, z_{n-2\lambda+2}, \ldots, z_{n-\lambda})$$

$$= L^0_{n-\lambda+1}(w)$$

$$+ \sum_{i=n-\lambda+2}^{n} \alpha^{i-(n-\lambda+1)} \left[\sum_{d=0}^{\infty} L^0_i \left(w + \sum_{j=n-2\lambda+2}^{i-\lambda} z_j - d \right) p_{n-\lambda+1, i-1}(d) \right]$$

$$+ \alpha^\lambda \sum_{d=0}^{\infty} t \left(w + \sum_{j=n-2\lambda+2}^{n-\lambda} z_j - d \right) p_{n-\lambda+1, n}(d), \qquad (11.9)$$

where $p_{i,j}(d)$ was defined at the end of Section 1. Assuming that no orders are outstanding at the beginning of period 1, then $w_1 = x_1$ and the minimum expected total discounted cost for the n-period problem is $f_1(x_1, 0, \ldots, 0)$.

Problem 11.2. Use (11.7)–(11.9) to find the optimal ordering policy and minimum expected cost for the following inventory problem:

$$n = 4, \quad \lambda = 2, \quad \alpha = 1, \quad x_1 = 0;$$
$$p(0) = \tfrac{3}{4}, \quad p(1) = \tfrac{1}{4};$$
$$c(0) = 0, \quad c(1) = 3, \quad c(z) = \infty \quad \text{for} \quad z \geqslant 2;$$
$$h(v) = v \quad \text{for} \quad v \geqslant 0;$$
$$\Pi(v) = 3v \quad \text{for} \quad v \geqslant 0;$$
$$s(v) = 2v \quad \text{for} \quad v \geqslant 0;$$
$$b(v) = 4v \quad \text{for} \quad v \geqslant 0.$$

The above formulation requires that for each i, $f_i(w, z_{i-\lambda+1}, \ldots, z_{i-1})$ be computed for all possible values of w, $z_{i-\lambda+1}, \ldots, z_{i-1}$. Therefore, if λ is very large, then this formulation is likely to be computationally intractable. We now consider a dynamic-programming formulation that requires one stage variable and only *one* state variable. Suppose that x items are on hand and on order at the beginning of period i. Let us make two observations. First, the number of items ordered in period i, say z, in no way affects the costs incurred in periods i through $i + \lambda - 1$. This is because the ordering cost $c_i(z)$ is incurred in period $i + \lambda$ and, furthermore, since the order for z items arrives in period $i + \lambda$, it does not affect the expected holding and shortage costs in periods i through $i + \lambda - 1$. For these reasons it is possible to neglect the costs in periods i through $i + \lambda - 1$ in the definition of an optimal expected value function. The second observation is that the amount of inventory actually on hand to satisfy demand in period $i + \lambda$, $w_{i+\lambda}$, depends only on x_i, the amount of inventory on hand and on order at the beginning of period i, on z_i, the order in period i (and not on previous orders), and on

$d_i, d_{i+1}, \ldots, d_{i+\lambda-1}$. In particular, $w_{i+\lambda}$ is given by

$$w_{i+\lambda} = y_i - \sum_{j=i}^{i+\lambda-1} d_j.$$

With these two observations in mind we define the optimal expected value function $f_i(x)$ as follows:

$f_i(x) =$ the expected total discounted cost (discounted to period $i + \lambda$) for periods $i + \lambda$ through n given that x items are on hand and on order at the beginning of period i and an optimal ordering policy is followed.   (11.10)

If we define $L_i(y)$ to be the expected holding and shortage cost in period $i + \lambda$ given that $y_i = y$, then $L_i(y)$ may be computed as follows:

$$L_i(y) = \sum_{d=0}^{\infty} \left[\sum_{d'=0}^{\infty} L_{i+\lambda}^0(y - d, d') p_{i+\lambda}(d') \right] p_{i, i+\lambda-1}(d)$$

$$= \sum_{d=0}^{\infty} L_{i+\lambda}^0(y - d) p_{i, i+\lambda-1}(d).$$

The appropriate recurrence relation for this formulation is

$$f_i(x) = \min_{y > x} \left[c_i(y - x) + L_i(y) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y - d) p_i(d) \right]$$

$$(i = 1, 2, \ldots, n - \lambda) \tag{11.11}$$

and the boundary condition is

$$f_{n-\lambda+1}(x) = \sum_{d=0}^{\infty} t(x - d) p_{n-\lambda+1, n}(d). \tag{11.12}$$

Notice that (11.11) is of the same form as (11.3) which is the recurrence relation for $\lambda = 0$. If we successively compute $f_{n-\lambda}$ through f_1, then $f_1(x_1)$ is the expected total discounted cost (discounted to period $\lambda + 1$) for periods $\lambda + 1$ through n. It does not include the expected holding and shortage costs for periods 1 through λ, which are not affected by the ordering decisions. The expected total discounted cost (discounted to period 1) for the entire n periods is then given by

$$L_1^0(x_1) + \sum_{i=2}^{\lambda} \left[\alpha^{i-1} \sum_{d=0}^{\infty} L_i^0(x_1 - d) p_{1, i-1}(d) \right] + \alpha^{\lambda} f_1(x_1),$$

where $L_i^0(w)$ is given by (11.1).

Let us now solve Problem 11.2 by the above procedure. The solution is as follows:

$$f_3(2) = \tfrac{9}{16} t(2) + \tfrac{6}{16} t(1) + \tfrac{1}{16} t(0) = -\tfrac{9}{4} - \tfrac{3}{4} - 0 = -3;$$

$$f_3(1) = \tfrac{9}{16} t(1) + \tfrac{6}{16} t(0) + \tfrac{1}{16} t(-1) = -\tfrac{9}{8} - 0 + \tfrac{1}{4} = -\tfrac{7}{8};$$

$$f_3(0) = 2;$$

$$f_3(-1) = 6;$$
$$f_3(-2) = 10.$$

$$f_2(1) = \min\left[0 + L_2(1) + \left\{\tfrac{3}{4}f_3(1) + \tfrac{1}{4}f_3(0)\right\}, 3 + L_2(2)\right.$$
$$\left. + \left\{\tfrac{3}{4}f_3(2) + \tfrac{1}{4}f_3(1)\right\}\right]$$
$$= \min\left[0 + \tfrac{15}{16} - \tfrac{5}{32}, 3 + \tfrac{21}{16} - \tfrac{79}{32}\right] = \min\left[\tfrac{25}{32}, 1\tfrac{3}{16}\right]$$
$$= \tfrac{25}{32}, \qquad y_2(1) = 1;$$
$$f_2(0) = 3\tfrac{25}{32}, \qquad y_2(0) = 1;$$
$$f_2(-1) = 8\tfrac{1}{4}, \qquad y_2(-1) = 0.$$

$$f_1(0) = \min\left[0 + L_1(0) + \left\{\tfrac{3}{4}f_2(0) + \tfrac{1}{4}f_2(-1)\right\}, 3 + L_1(1)\right.$$
$$\left. + \left\{\tfrac{3}{4}f_2(1) + \tfrac{1}{4}f_2(0)\right\}\right]$$
$$= \min\left[0 + \tfrac{9}{4} + \tfrac{627}{128}, 3 + \tfrac{15}{16} + \tfrac{196}{128}\right] = \min\left[7\tfrac{19}{128}, 5\tfrac{15}{32}\right]$$
$$= 5\tfrac{15}{32}, \qquad y_1(0) = 1.$$

Therefore, the minimum expected cost for periods 3 and 4 is $5\tfrac{15}{32}$ and the minimum expected cost for periods 1 through 4 is

$$L_1^0(0) + \left[\tfrac{3}{4}L_2^0(0) + \tfrac{1}{4}L_2^0(-1)\right] + f_1(0) = \tfrac{3}{4} + \tfrac{3}{2} + 5\tfrac{15}{32} = 7\tfrac{23}{32}.$$

This is the same answer that we got for Problem 11.2, as it should be if both formulations are correct.

The above formulation depended on $c_i(z)$ being incurred in period $i + \lambda$. However, it is possible to give a similar formulation when $c_i(z)$ is incurred in period i.

Let us now consider the lost sales case for $\lambda \geqslant 2$. As you might expect, it is still possible to give a dynamic-programming formulation with one stage variable and λ state variables. But once again this solution is likely to be computationally intractable. Therefore, the question naturally arises as to whether it is possible to give a formulation (similar to the backlogged case) that requires only one state variable. Unfortunately, the answer is no. To see why, consider the case $\lambda = 2$. In the backlogged case, the amount of inventory on hand to meet demand in period $i + 2$, w_{i+2}, is given by

$$w_{i+2} = x_i + z_i - (d_i + d_{i+1}).$$

Thus, given x_i, w_{i+2} depends only on z_i and the sum of d_i and d_{i+1}. However, in the lost sales case w_{i+2} is given by

$$w_{i+2} = \max\left[w_{i+1} - d_{i+1}, 0\right] + z_i$$
$$= \max\left[\max\{w_i - d_i, 0\} + z_{i-1} - d_{i+1}, 0\right] + z_i$$
$$= \max\left[\max\{x_i - z_{i-1} - d_i, 0\} + z_{i-1} - d_{i+1}, 0\right] + z_i.$$

In general, the two expressions for w_{i+2} will not be the same. Furthermore, w_{i+2} can no longer be computed given only x_i. Therefore, it is not possible

to give a formulation that requires only one state variable. For this reason we shall restrict our attention to the backlogged case in most of the remainder of this book. Also, it was with the backlogged case in mind that we defined (in Section 1) x_i to be the amount of inventory on hand and on order at the beginning of period i.

4. A Model with Uncertain Delivery Lag

Up to now we have always assumed that the delivery lag λ is a known nonnegative integer. We now consider the case where λ is a random variable that takes on the value k with probability $q(k)$ for $k = 1, 2, \ldots, m$. If z items are ordered in period i, then $c_i(z)$ is incurred at the time of ordering. Items that arrive after period $n + 1$ are assumed to have no salvage value. To simplify matters, it is also assumed that there can never be more than one order outstanding at a time (see Problem 11.3 for a different assumption). Furthermore, we must now assume that the delivery of any previous order in period i precedes the ordering decision in period i. (Otherwise, the company may not know when it is allowed to order.) The sequence of events (during a time period) for the model of *this section* is shown in Figure 11.2.

If w items are available to meet demand in period i, define $L_{i,j}^0(w)$ to be the expected total discounted holding and shortage costs (discounted to period i) in periods $i, i + 1, \ldots, j$ given that no orders arrive in periods $i + 1, i + 2, \ldots, j$. (If an order arrived at the beginning of period i, then it is included in w.) Then $L_{i,j}^0(w)$ is given by

$$L_{i,j}^0(w) = L_i^0(w) + \sum_{l=i+1}^{j} \alpha^{l-i} \left[\sum_{d=0}^{\infty} L_l^0(w - d)p_{i, l-1}(d) \right].$$

We now give the dynamic-programming formulation for this problem. Define the optimal expected value function $f_i(w)$ as follows:

<div align="center">

Period i

Inventory on hand
</div>

w_i		$w_i \cdot d_i$
delivery	order z_i	demand d_i
x_i	y_i	$y_i \cdot d_i$

<div align="center">

Inventory on hand and on order
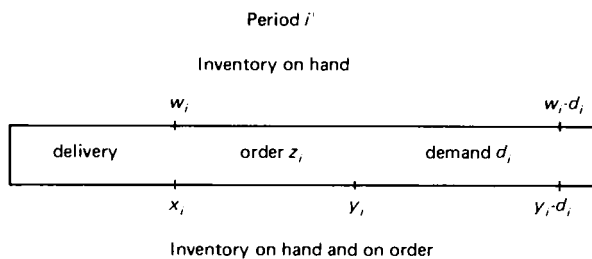
Figure 11.2
</div>

$f_i(w)$ = the expected total discounted cost (discounted to period i) for periods i through n given that w items are on hand in period i (after an order, if any, has been delivered), there is not an outstanding order, and an optimal ordering policy is followed.   (11.13)

The appropriate recurrence relations are

$$f_i(w) = \min \left[\begin{array}{l} c_i(0) + L_i^0(w) + \alpha \sum_{d=0}^{\infty} f_{i+1}(w-d)p_i(d) \\[2ex] \min_{y>w} \left\{ c_i(y-w) + \sum_{k=1}^{m} \left[L_{i,\,i+k-1}^0(w) \right. \right. \\[2ex] \left. \left. + \alpha^k \sum_{d=0}^{\infty} f_{i+k}(y-d)p_{i,\,i+k-1}(d) \right] q(k) \right\} \end{array} \right]$$

$(i = 1, 2, \ldots, n - m);$ (11.14)

$$f_i(w) = \min \left[\begin{array}{l} c_i(0) + L_i^0(w) + \alpha \sum_{d=0}^{\infty} f_{i+1}(w-d)p_i(d) \\[2ex] \min_{y>w} \left\{ c_i(y-w) + \sum_{k=1}^{n-i} \left[L_{i,\,i+k-1}^0(w) \right. \right. \\[2ex] \left. + \alpha^k \sum_{d=0}^{\infty} f_{i+k}(y-d)p_{i,\,i+k-1}(d) \right] q(k) \\[2ex] + \left[L_{i,\,n}^0(w) + \alpha^{n-i+1} \sum_{d=0}^{\infty} t(y-d)p_{i,\,n}(d) \right] q(n-i+1) \\[2ex] \left. + \sum_{k=n-i+2}^{m} \left[L_{i,\,n}^0(w) + \alpha^{n-i+1} \sum_{d=0}^{\infty} t(w-d)p_{i,\,n}(d) \right] q(k) \right\} \end{array} \right]$$

$(i = n - m + 1, n - m + 2, \ldots, n - 1).$

The top line on the r.h.s. of each recurrence relation represents the expected cost if zero items are ordered in period i. The bottom line of the first recurrence relation represents the expected cost when $y - w$ items are ordered. If the delivery lag is k periods, it includes a term for the expected holding and shortage costs in periods i through $i + k - 1$ and a term for the expected cost for periods $i + k$ (the time of delivery) through n. The bottom line of the second recurrence relation has a similar interpretation.

The boundary condition for this problem is

$$f_n(w) = L_n^0(w) + \alpha \sum_{d=0}^{\infty} t(w - d)p_n(d). \qquad (11.15)$$

Assuming that an order is not outstanding at the beginning of period 1, then $w_1 = x_1$ and the answer is $f_1(x_1)$.

Problem 11.3. Suppose that in the inventory model of Section 4 there can be more than one order outstanding. Assume, however, that an order placed in period i is never delivered before an order placed in an earlier period. (It may be delivered in the same period.) Let $r(k)$ be the probability that *all* orders placed k or more periods ago and not yet delivered arrive in period i. Assume that $r(k)$ is independent of the number of orders not yet delivered and also their sizes. Give a dynamic-programming formulation.

In the following problems assume that $\lambda = 0$.

Problem 11.4. (Production Scheduling with Smoothing Costs) Consider a company that manufactures its own items and would like to schedule its production for the next n periods. Suppose that in addition to the normal ordering, holding, and shortage costs there is a smoothing cost $a_i(z_{i-1}, z)$ which results from changing the production level from z_{i-1} in period $i - 1$ to z in period i. Assume that the production level is zero before the planning horizon begins. Give a dynamic-programming formulation.

Problem 11.5. (Maximizing Profit) Suppose that a company receives a revenue r for each item it sells, and that the revenue is remitted to the company when the item is delivered to the customer. Give a dynamic-programming formulation that the company can use to maximize its expected total discounted profit.

Problem 11.6. (Perishable Inventory) Suppose that due to deterioration an item delivered in period i cannot be used to meet demand after period $i + 2$. Give a dynamic-programming formulation to minimize the expected total discounted cost.

Problem 11.7. (Uncertain Planning Horizon) Suppose that the planning horizon is a random variable because the product will become obsolete at some uncertain time in the future. In particular, let $r(k)$ be the probability at the start of period 1 that the process terminates at the end of period k, for $k = 1, 2, \ldots, n$. The company finds out after the completion of period k whether the process terminates or continues at least one more period. Give a dynamic-programming formulation. (For a similar path problem, see Problem 9.6.)

Problem 11.8. (Cash Management) A mutual fund would like to decide how much cash to have in its bank account for each of the next n time periods. At the beginning of each period, the cash balance can be increased by selling stocks

(at a cost of c_s per dollar value of the stocks), can be decreased by buying stocks (at a cost of c_b per dollar value of the stocks), or can be left constant. Assume that the amount of time required to implement the decision is negligible. During the period (after the decision has been implemented) the mutual fund receives demands for cash from customers redeeming their mutual fund shares and cash inflows from the sale of shares. Let $p_i(d)$ be the probability that the cash balance changes by an amount d during period i. Note that d may be positive or negative. If during a period the cash balance falls below zero, then the bank will automatically lend the fund the additional amount. However, the fund must pay the bank an interest charge of c_p per dollar per period (based on the cash balance at the end of period). Conversely, if the fund has a positive cash balance at the end of a period, then it incurs a cost of c_h per dollar per period due to the fact that the fund's money could have been invested elsewhere. Assuming that there are no terminal costs, give a dynamic-programming formulation to minimize the expected total discounted cost for operating the fund.

INVENTORY MODELS WITH SPECIAL COST ASSUMPTIONS

1. Introduction

In Chapter 11 we described the general nature of an inventory system and, furthermore, we showed how dynamic programming could be used to compute an optimal ordering policy. For each inventory model that we considered, the optimal policy was specified by stating how many items to order for each possible value of the state variable (usually the number of items on hand and on order). Thus, a large amount of data was necessary to specify the optimal policy.

We shall now see that by making special assumptions about the ordering, holding, and penalty costs, it is possible to characterize the *form* of an optimal policy. This will in turn reduce both the number of calculations required to get a solution and the amount of data required to specify the solution once it has been determined. Furthermore, for several of the inventory models that we consider, dynamic programming will actually be used to prove the optimality of a particular type of policy.

We shall find it mathematically convenient in this chapter to treat all variables as being continuous. However, this will not affect the validity of the models that we consider (see example in Section 3 and comment at the end of Section 4).

2. Convex and Concave Cost Functions

This section will be devoted to defining and stating properties of convex and concave functions. We shall make use of these results in Sections 3–5 of this chapter. In what follows, all sets are considered to be subsets of E^n, n-dimensional Euclidian space for some $n \geqslant 1$.

156

A set S is *convex* if $x, y \in S$ and $0 \leqslant \alpha \leqslant 1$ imply that $\alpha x + (1 - \alpha)y \in S$. Thus, a set S is convex if all points on the line segment joining two points in S are also in S. Examples of convex sets are E^n, $[0, \infty)$, and the set of solutions to a system of linear equations and inequalities.

Let $x, x^1, x^2, \ldots, x^k \in E^n$ and let $\lambda^1, \lambda^2, \ldots, \lambda^k$ be nonnegative numbers such that $\sum_{i=1}^{k} \lambda^i = 1$. If $x = \sum_{i=1}^{k} \lambda^i x^i$, then x is a *convex combination* of $x^1, x^2, \ldots, x^k$.

Problem 12.1. Prove that if S is a convex set and x is a convex combination of points in S, then $x \in S$.

If S is a convex set and $x \in S$, then x is an *extreme point* of S if there do not exist $x^1, x^2 \in S$ ($x^1 \neq x^2$) and $0 < \alpha < 1$ such that

$$x = \alpha x^1 + (1 - \alpha)x^2.$$

Thus, x is an extreme point of a convex set S if it is an element of S and if it does not lie in the interior of any line segment in S. Let A be an $m \times n$ matrix, $x \in E^n$, and $b \in E^m$.

Lemma 12.1. If $S = \{x : Ax = b, x \geqslant 0\}$ and S is nonempty and bounded, then S has a finite number of extreme points. Furthermore, every point in S is a convex combination of these extreme points. (For a proof, see Luenberger [2, pp. 20–22]. A similar result is true if $Ax = b$ is replaced by a system of linear equations and inequalities.)

A real-valued function $f(x)$ defined on a convex set S is *convex* if $f(\alpha x + (1 - \alpha)y) \leqslant \alpha f(x) + (1 - \alpha)f(y)$ for all $x, y \in S$ and $0 \leqslant \alpha \leqslant 1$. If $-f(x)$ is convex, then $f(x)$ is *concave*. Note that the sum of convex (concave) functions is convex (concave) and also that a linear function is both convex and concave. Several other examples of convex and concave functions are given in Figure 12.1. The last function would represent the ordering cost when the cost of ordering z items is the sum of a set-up cost K plus a cost of c units per item.

Problem 12.2. Let S be a convex set, $x^1, x^2, \ldots, x^k \in S$, and $\lambda^1, \lambda^2, \ldots, \lambda^k$ nonnegative numbers such that $\sum_{i=1}^{k} \lambda^i = 1$. Prove that if $f(x)$ is concave on S, then

$$f\left(\sum_{i=1}^{k} \lambda^i x^i\right) \geqslant \sum_{i=1}^{k} \lambda^i f(x^i).$$

Theorem 12.2. Suppose that $S = \{x : Ax = b, x \geqslant 0\}$ and S is nonempty and bounded. If $f(x)$ is concave on S, then $f(x)$ attains its minimum over S at an extreme point of S.

Proof: Let x be an arbitrary point in S. From Lemma 12.1, S has finitely many extreme points $x^1, x^2, \ldots, x^k$ and there exist nonnegative
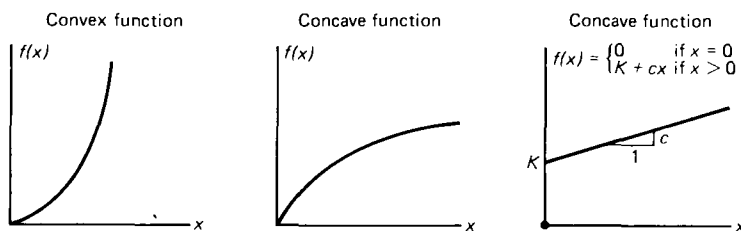
Figure 12.1

numbers $\lambda^1, \lambda^2, \ldots, \lambda^k$ such that $\sum_{i=1}^{k} \lambda^i = 1$ and $x = \sum_{i=1}^{k} \lambda^i x^i$. Let x^j be an extreme point such that $f(x^j) \leqslant f(x^i)$ for $i = 1, 2, \ldots, k$. Then

$$f(x) = f\left(\sum_{i=1}^{k} \lambda^i x^i\right) \geqslant \sum_{i=1}^{k} \lambda^i f(x^i) \geqslant \sum_{i=1}^{k} \lambda^i f(x^j) = f(x^j).$$

It follows that f attains its minimum at the extreme point x^j. ∎

A real-valued function $f(x)$ is K-*convex* $(K \geqslant 0)$ on E^1 if

$$K + f(x + a) - f(x) - \frac{a}{b}\left[f(x) - f(x - b)\right] \geqslant 0$$

for all $x \in E^1$ and all positive numbers a, b. Thus, if f is a K-convex function, then the linear approximation to f that equals f at x and $x - b$ does not lie above f at $x + a$ by more than K. An example of a K-convex function is given in Figure 12.2.

Problem 12.3. Prove the following properties of a K-convex function:

(a) f is 0-convex if and only if f is convex;
(b) if f is K-convex, then f is M-convex for all $M \geqslant K$;
(c) if f and g are K-convex and M-convex, respectively, and $\alpha, \beta > 0$, then $\alpha f + \beta g$ is $(\alpha K + \beta M)$-convex;
(d) if $f(x)$ is K-convex, then $g(x) \equiv f(x - z)$ is K-convex for all $z \in E^1$;
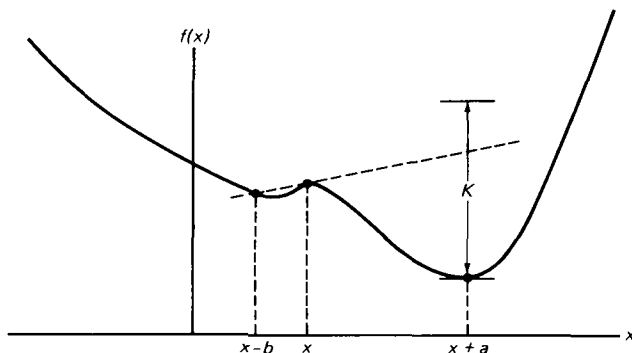
Figure 12.2

(e)   if $f(x)$ is K-convex and $\phi(z)$ is a density function, then

$$g(x) \equiv \int_0^\infty f(x - z)\phi(z) \, dz$$

is K-convex.

3. Models with Deterministic Demand and Concave Costs

Consider an n-period inventory problem with $\lambda = 0$ and $\alpha = 1$. (If a discount factor less than 1 is desired, then it is sufficient to multiply all cost functions in period i by α^{i-1}.) Let d_i be the known demand in period i, z_i the amount produced (or ordered) in period i, and v_i the inventory level at the *end* of period i. It is assumed that the inventory level at the beginning of period 1 is zero ($v_0 = 0$), that the final inventory must be zero ($v_n = 0$), and that all demand must be satisfied on time ($v_i \geq 0$ for $i = 1, 2, \ldots, n$). It follows that

$$v_i = \sum_{j=1}^i z_j - \sum_{j=1}^i d_j \quad \text{for} \quad i = 1, 2, \ldots, n.$$

Let $z = (z_1, z_2, \ldots, z_n)$. Then the set of all feasible production policies is given by

$$S = \left\{ z : z_i \geq 0 \quad \text{and} \quad \sum_{j=1}^i z_j - \sum_{j=1}^i d_j \geq 0 \quad \text{for} \quad i = 1, 2, \ldots, n; \right.$$

$$\left. \sum_{j=1}^n z_j - \sum_{j=1}^n d_j = 0 \right\}.$$

Let $c_i(z_i)$ be the cost of producing z_i items in period i and let $h_i(v_i)$ be the holding cost in period i when the inventory level at the end of period i is v_i. (There are no shortage costs since $v_i \geq 0$ for all i.) For any $z \in S$, define the function $g(z)$ by

$$g(z) = \sum_{i=1}^n c_i(z_i) + \sum_{i=1}^n h_i \left(\sum_{j=1}^i (z_j - d_j) \right).$$

(All functions considered in this section, with the exception of $g(z)$, have a scalar as their argument.) Thus, $g(z)$ is the total cost for the n-period problem when the policy z is employed. The objective of the company is to find a policy $z \in S$ whose corresponding cost, $g(z)$, is minimal.

We shall begin by giving the standard forward dynamic-programming formulation for this problem. (We have not yet made any assumptions about $c_i(z_i)$ or $h_i(v_i)$.) Define the optimal value function $f_i(v)$ as

$f_i(v) =$ the minimum cost for periods 1 through i given that we must have an inventory of v items at the end of period i.   (12.1)

Then $f_i(v)$ satisfies the recurrence relation

$$f_i(v) = \min_{0 \leqslant z \leqslant v + d_i} \left[c_i(z) + h_i(v) + f_{i-1}(v - z + d_i) \right] \qquad (i = 2, 3, \ldots, n)$$

(12.2)

and the boundary condition is

$$f_1(v) = c_1(v + d_1) + h_1(v). \tag{12.3}$$

For each value of i, $f_i(v)$ is computed for all values of v between 0 and $\sum_{j=i+1}^{n} d_j$. The answer is $f_n(0)$. (Since $v_n = 0$, there are no terminal costs.)

Let us henceforth assume that each $c_i(z_i)$ and each $h_i(v_i)$ are concave on the set $[0, \infty)$. These assumptions will allow us to characterize the form of an optimal policy. Our presentation in the remainder of this section will follow that of Topkis [4]. However, for some of the original proofs and also additional results, see Eppen *et al.* [1], Wagner and Whitin [6], and Zangwill [8, 9].

The function $g(z)$ is now concave on the set S which is nonempty and bounded. It follows from Theorem 12.2 that there exists an optimal policy that is an extreme point of S. A policy $z \in S$ is said to have the *regeneration point property* (abbreviated r.p.p.) if for each i, either $v_{i-1} = 0$ or $z_i = 0$ or both; i.e., if $v_{i-1}z_i = 0$ for each i. For a policy $z \in S$, we shall say that period i is a *regeneration point* (abbreviated r.p.) if $v_i = 0$. (This terminology arises since if $v_i = 0$ in some optimal policy, then the situation at the beginning of period $i + 1$ is essentially the same as it was at the beginning of period 1. For a similar idea, see Chapter 2, Section 4.) Note that periods 0 and n are r.p.'s. Let u^i denote the ith unit vector (a vector with a 1 as its ith component and 0 elsewhere). The following theorem shows that there exists an optimal policy with the r.p.p.

Theorem 12.3. Every extreme point of S has the r.p.p. Therefore, there is an optimal policy with the r.p.p.

Proof: Suppose that z is an extreme point without the r.p.p. Let j be the smallest integer such that $v_{j-1}z_j > 0$ and let k be the largest integer less than j such that $z_k > 0$ (v_{j-1} is produced in period k). For $\epsilon > 0$, let $z^+(\epsilon) = z + \epsilon(u^j - u^k)$ and $z^-(\epsilon) = z - \epsilon(u^j - u^k)$. If we let $\bar{\epsilon} = \min[v_{j-1}, z_j]$, then it is easy to show that $z^+(\bar{\epsilon})$, $z^-(\bar{\epsilon}) \in S$. Furthermore, $z = \frac{1}{2}z^+(\bar{\epsilon}) + \frac{1}{2}z^-(\bar{\epsilon})$, which contradicts the fact that z is an extreme point. Therefore, z must have the r.p.p. ∎

For $j \leqslant i$, let $c(j, i)$ be the total cost in periods j through i given that periods $j - 1$ and i are r.p.'s and the only production during periods j through i is in period j. Then $c(j, i)$ is given by

$$c(j, i) = c_j \left(\sum_{s=j}^{i} d_s \right) + \sum_{s=j+1}^{i} c_s(0) + \sum_{s=j}^{i-1} h_s \left(\sum_{t=s+1}^{i} d_t \right).$$

The above characterization of an optimal policy can be used to obtain a dynamic-programming formulation which is more efficient than the one given before. Define the optimal value function f_i as follows:

f_i = the minimum cost for periods 1 through i given that we
  must have an inventory of 0 items at the end of period i.   (12.4)

Then f_i satisfies the recurrence relation

$$f_i = \min_{1 \leqslant j \leqslant i} \left[f_{j-1} + c(j, i) \right] \qquad (i = 1, 2, \ldots, n) \qquad (12.5)$$

and the boundary condition is

$$f_0 = 0. \qquad (12.6)$$

Notice that (12.5) is also the recurrence relation for finding a shortest path in an acyclic network. In addition, notice that $f_i = f_i(0)$, where $f_i(0)$ was defined by (12.1). Thus, by knowing the form of an optimal policy, we are able to avoid having to compute $f_i(v)$ for $v > 0$.

We shall now show, using mathematical induction, that f_i is correctly given by (12.5) and (12.6). The case $i = 1$ is obvious. Therefore, assume that $f_1, f_2, \ldots, f_{i-1}$ are correctly given by (12.5) and (12.6). By Theorem 12.3, we can restrict our attention to policies with the r.p.p. in computing f_i. For an i-period problem with $v_i = 0$, consider all policies with the r.p.p. for which the last production occurs in period j ($1 \leqslant j \leqslant i$). It follows from our inductive hypothesis that $f_{j-1} + c(j, i)$ is the minimum cost for this type of policy. (There is a policy with the r.p.p. whose cost for periods 1 through $j - 1$ is f_{j-1}.) Since we are free to choose j optimally, it follows that f_i is correctly given by (12.5).

In order to determine an optimal policy, we successively compute $f_1, f_2, \ldots, f_n$. The minimum cost for the n-period problem is, of course, f_n. Let $j(n)$ be a value of j that minimizes the r.h.s. of (12.5) when computing f_n. Then in some optimal policy,

$$z_{j(n)} = \sum_{s=j(n)}^{n} d_s, \qquad z_s = 0 \qquad \text{for} \quad s = j(n) + 1, \ldots, n,$$

and period $j(n) - 1$ is a r.p. To find an optimal policy for periods 1 through $j(n) - 1$, let us denote $j(n) - 1$ by m and let $j(m)$ be a value of j that minimizes the r.h.s. of (12.5) when computing f_m. Then in some optimal policy,

$$z_{j(m)} = \sum_{s=j(m)}^{m} d_s, \qquad z_s = 0 \qquad \text{for} \quad s = j(m) + 1, \ldots, m,$$

and period $j(m) - 1$ is a r.p., etc.

Let us now consider an example. Suppose we have a four-period inventory problem with the following data ($c(z)$ is assumed to be a concave function but we give its value only for relevant z):

$$d_1 = 1, \quad d_2 = 2, \quad d_3 = 2, \quad d_4 = 1;$$
$$c(0) = 0, \quad c(1) = 5, \quad c(2) = 9, \quad c(3) = 12, \quad c(4) = 14,$$
$$c(5) = 16, \quad c(6) = 18;$$
$$h(v) = v \quad \text{for} \quad v \geqslant 0.$$

The solution to this problem is as follows:

$$c(1, 1) = c(1) = 5,$$
$$c(1, 2) = c(3) + h(2) = 12 + 2 = 14,$$
$$c(1, 3) = c(5) + h(4) + h(2) = 16 + 4 + 2 = 22,$$
$$c(1, 4) = c(6) + h(5) + h(3) + h(1) = 18 + 5 + 3 + 1 = 27,$$
$$c(2, 2) = c(2) = 9,$$
$$c(2, 3) = c(4) + h(2) = 14 + 2 = 16,$$
$$c(2, 4) = c(5) + h(3) + h(1) = 16 + 3 + 1 = 20,$$
$$c(3, 3) = c(2) = 9,$$
$$c(3, 4) = c(3) + h(1) = 12 + 1 = 13,$$
$$c(4, 4) = c(1) = 5.$$

$$f_0 = 0;$$
$$f_1 = f_0 + c(1, 1) = 0 + 5 = 5, \qquad j(1) = 1;$$
$$f_2 = \min[\, f_0 + c(1, 2), f_1 + c(2, 2)\,]$$
$$\quad = \min[\, 0 + 14, 5 + 9\,] = 14, \qquad j(2) = 1 \text{ or } 2;$$
$$f_3 = \min[\, f_0 + c(1, 3), f_1 + c(2, 3), f_2 + c(3, 3)\,]$$
$$\quad = \min[\, 0 + 22, 5 + 16, 14 + 9\,] = 21, \qquad j(3) = 2;$$
$$f_4 = \min[\, f_0 + c(1, 4), f_1 + c(2, 4), f_2 + c(3, 4), f_3 + c(4, 4)\,]$$
$$\quad = \min[\, 0 + 27, 5 + 20, 14 + 13, 21 + 5\,] = 25, \qquad j(4) = 2.$$

Therefore, periods 0, 1, and 4 are r.p.'s and an optimal production policy is $z_1 = 1$, $z_2 = 5$, $z_3 = 0$, $z_4 = 0$.

Up to now we have assumed that all demand must be satisfied on time. Let us now consider a similar inventory model where backlogging is allowed. If v_i is negative, then there is a backlog of $-v_i$ items at the end of period i. Let $l_i(v_i)$ be the holding and penalty cost in period i when the net inventory level at the end of period i is v_i. Then $l_i(v_i)$ is given by

$$l_i(v_i) = \begin{bmatrix} h_i(v_i) & \text{if} \quad v_i \geqslant 0, \\ \Pi_i(-v_i) + h_i(0) & \text{if} \quad v_i \leqslant 0. \end{bmatrix}$$

We shall also assume that $c_i(z_i)$ is concave on $[0, \infty)$, that $h_i(v_i)$ is concave on $[0, \infty)$, and that $\Pi_i(-v_i)$ is concave on $[0, \infty)$. Thus, $l_i(v_i)$ is concave on $[0, \infty)$ and also on $(-\infty, 0]$. However, it may not be concave on $(-\infty, \infty)$. An example of the function $l_i(v_i)$ is given in Figure 12.3.
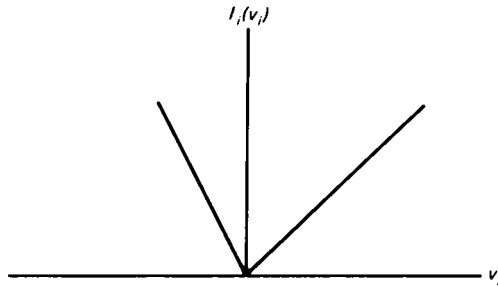
Figure 12.3

The set of feasible production policies is now given by

$$S = \left\{ z : z_i \geqslant 0 \quad \text{for} \quad i = 1, 2, \ldots, n; \ \sum_{j=1}^{n} z_j - \sum_{j=1}^{n} d_j = 0 \right\}$$

and the objective of the company is to minimize

$$g(z) = \sum_{i=1}^{n} c_i(z_i) + \sum_{i=1}^{n} l_i \left(\sum_{j=1}^{i} (z_j - d_j) \right)$$

over $z \in S$.

A set of the form

$$B_m = \left\{ z : z \in S, (-1)^{m_i} v_i \geqslant 0 \quad \text{for} \quad i = 1, 2, \ldots, n - 1 \right\},$$

where m is an $(n - 1)$-vector with ith component m_i equal to either 0 or 1, will be called a *basic set*. There are 2^{n-1} basic sets, each of which is nonempty and bounded, and S is the union of the basic sets. Furthermore, $g(z)$ is concave on each basic set. It follows from Theorem 12.2 that $g(z)$ attains its minimum over a particular basic set at an extreme point of that basic set. Therefore, there exists an optimal policy that is an extreme point of some basic set.

A policy $z \in S$ will now be said to have the *regeneration point property* (r.p.p.) if whenever $i < k$, $z_i > 0$, $z_k > 0$, and $z_j = 0$ for $i < j < k$, then there exists an l such that $i \leqslant l \leqslant k - 1$ and $v_l = 0$. For $z \in S$, we shall still say that period l is a regeneration point (r.p.) of $v_l = 0$. The following theorem shows that there exists an optimal policy with the r.p.p.

Theorem 12.4. Each extreme point of each basic set has the r.p.p. Therefore, there exists an optimal policy with the r.p.p.

Proof: Suppose that z is an extreme point of some basic set B_m that does not have the r.p.p. Then there exist integers i and k such that $i < k$, $z_i > 0$, $z_k > 0$, $z_j = 0$ for $i < j < k$, and $v_l \neq 0$ for $i \leqslant l \leqslant k - 1$. For $\epsilon > 0$, let $z^+ (\epsilon) = z + \epsilon(u^k - u^i)$ and $z^- (\epsilon) = z - \epsilon(u^k - u^i)$. If we let $\bar{\epsilon} = \min[z_i, z_k, |v_i|, \ldots, |v_{k-1}|]$, then $z^+ (\bar{\epsilon}), z^- (\bar{\epsilon}) \in B_m$. Furthermore,

$z = \frac{1}{2} z^{+}(\bar{\epsilon}) + \frac{1}{2} z^{-}(\bar{\epsilon})$, which contradicts the fact that z is an extreme point of B_m. Therefore, z must have the r.p.p. ∎

For $j \le k \le i$, let $c^k(j, i)$ be the total cost in periods j through i given that periods $j - 1$ and i are r.p.'s and the only production during periods j through i is in period k. Then $c^k(j, i)$ is given by

$$c^k(j, i) = c_k\left(\sum_{s=j}^{i} d_s\right) + \sum_{\substack{s=j \\ s \ne k}}^{i} c_s(0) + \sum_{s=j}^{k-1} l_s\left(-\sum_{t=j}^{s} d_t\right) + \sum_{s=k}^{i-1} l_s\left(\sum_{t=s+1}^{i} d_t\right).$$

Furthermore, let

$$c(j, i) = \min_{j \le k \le i} c^k(j, i).$$

It follows that $c(j, i)$ is the minimum total cost for periods j through i given that periods $j - 1$ and i are r.p.'s and only one production is allowed during periods j through i.

We now give the dynamic-programming formulation for the backlogged case. Let f_i be as defined by (12.4). Then f_i satisfies the same recurrence relation and boundary condition as before. Namely,

$$f_i = \min_{1 \le j \le i} \left[f_{j-1} + c(j, i) \right] \qquad (i = 1, 2, \ldots, n) \qquad (12.7)$$

and

$$f_0 = 0. \qquad (12.8)$$

Let us show that f_i is correctly given by (12.7) and (12.8) in the backlogged case. We shall use induction, but we give only the inductive step. Suppose that $f_1, f_2, \ldots, f_{i-1}$ are correctly given by (12.7) and (12.8). By Theorem 12.4, we can restrict our attention to policies with the r.p.p. in computing f_i. For an i-period problem with $v_i = 0$, consider all policies with the r.p.p. such that period $j - 1$ is the last r.p. before period i. It follows from Theorem 12.4 that only one production can occur during periods j through i. Therefore, by our inductive hypothesis, $f_{j-1} + c(j, i)$ is the minimum cost for the specified type of policy. Since we are free to choose j optimally, it follows that f_i is correctly given by (12.7).

In order to determine an optimal policy, we successively compute $f_1, f_2, \ldots, f_n$. Let $j(n)$ be a value of j that minimizes the r.h.s. of (12.7) when computing f_n and suppose that $c(j(n), n) = c^k(j(n), n)$. Then period $j(n) - 1$ is a r.p. and in some optimal policy $z_k = \sum_{s=j(n)}^{n} d_s$ and $z_s = 0$ for $s = j(n), \ldots, k - 1, k + 1, \ldots, n$. An optimal policy is now determined for periods 1 through $j(n) - 1$, etc.

Problem 12.4. Solve the previous example in the backlogged case. Assume now that $h(v) = v$ for $v \ge 0$ and $\Pi(v) = 2v$ for $v \ge 0$.

We conclude this section by showing that under certain conditions, the above dynamic-programming formulations can be made even more

efficient. For either of the two models, now let $j(i)$ be the *largest* integer j such that $j \leqslant i$ and $f_i = f_{j-1} + c(j, i)$. Also, if for all i, $c(j, i) - c(j, i - 1)$ is nonincreasing in j for $j \leqslant i - 1$, then we shall say that *condition M* (for monotone) holds.

Theorem 12.5. If condition M holds, then $j(i)$ is nondecreasing in i. Therefore, recurrence relations (12.5) and (12.7) may be replaced by

$$f_i = \min_{j(i-1) < j \leqslant i} \left[f_{j-1} + c(j, i) \right] \qquad (i = 1, 2, \ldots, n), \qquad (12.9)$$

which is a computational improvement.

Proof: We can write f_i as

$$f_i = \min_{1 < j \leqslant i} \left[f_{j-1} + c(j, i) \right]$$

$$= \min_{1 < j \leqslant i} \left[f_{j-1} + c(j, i - 1) + c(j, i) - c(j, i - 1) \right].$$

Suppose that $j(i - 1) = k$. Then for $j < k$ we have

$$f_{j-1} + c(j, i - 1) \geqslant f_{k-1} + c(k, i - 1)$$

and

$$c(j, i) - c(j, i - 1) \geqslant c(k, i) - c(k, i - 1),$$

where the last inequality follows from condition M. It follows that

$$f_{j-1} + c(j, i) \geqslant f_{k-1} + c(k, i) \qquad \text{for all} \quad j < k.$$

Therefore, when computing f_i, the minimization does not need to include $j < k$. In otherwords, $j(i) \geqslant j(i - 1)$. ∎

Problem 12.5. In the no-backlog model suppose that

$$c_i(z_i) = \begin{bmatrix} 0 & \text{if} \quad z_i = 0 \\ K_i + c_i z_i & \text{if} \quad z_i > 0 \end{bmatrix}$$

(see Figure 12.1) and $h_i(v_i) = h_i v_i$. If $c_j \leqslant c_i + \sum_{l=i}^{j-1} h_l$ for all $i < j$, or equivalently, $c_{i+1} \leqslant c_i + h_i$ for all i, then show that condition M holds.

4. Optimality of (s, S) Policies

Consider the n-period inventory model of Chapter 11, Section 3. Assume that the excess of demand over supply at the end of a period is backlogged. Also assume that the demand in period i is a continuous random variable with density function $\phi_i(z)$. If $L_i(y)$ is the expected holding and shortage cost in period $i + \lambda$ given that $y_i = y$, then $L_i(y)$ may now be computed as

$$L_i(y) = \int_0^\infty \left[\int_0^\infty L_{i+\lambda}^0(y - z, z') \phi_{i+\lambda}(z') dz' \right] \phi_{i, i+\lambda-1}(z) \, dz$$

$$= \int_0^\infty L_{i+\lambda}^0(y - z) \phi_{i, i+\lambda-1}(z) \, dz,$$

where $\phi_{i,\,i+\lambda-1}(z)$ is the density function for the total demand in periods $i, i + 1, \ldots, i + \lambda - 1$. Let $f_i(x)$ be as defined by (11.10). Then $f_i(x)$ satisfies the following recurrence relation and boundary condition:

$$f_i(x) = \min_{y > x} \left[c_i(y - x) + L_i(y) + \alpha \int_0^\infty f_{i+1}(y - z)\phi_i(z)\, dz \right]$$
$$(i = 1, 2, \ldots, n - \lambda), \tag{12.10}$$

$$f_{n-\lambda+1}(x) = \int_0^\infty t(x - z)\phi_{n-\lambda+1,\,n}(z)\, dz. \tag{12.11}$$

Since our primary objective in this section is to characterize the form of an optimal ordering policy, we shall neglect the expected holding and shortage costs in periods 1 through λ, which are unaffected by the ordering decisions. Therefore, we shall restrict our attention to $f_1(x_1)$ which is the expected total discounted cost (discounted to period $\lambda + 1$) for periods $\lambda + 1$ through n. Note that multiplying $f_1(x_1)$ by α^λ to discount to period 1 will not affect the form of an optimal policy.

Let us assume that $c_i(z)$ is now given by

$$c_i(z) = K_i \delta(z) + c_i z \qquad \text{for} \quad i = 1, 2, \ldots, n - \lambda,$$

where $\delta(0) = 0$, $\delta(z) = 1$ for $z > 0$, $K_i \geq 0$, and $c_i \geq 0$. Thus, the cost of ordering z items in period i is the sum of a set-up cost K_i and a cost that is a linear function of z. Also assume that $K_{i-1} \geq \alpha K_i$, that $L_i(y)$ is convex ($h_{i+\lambda}(y) + \Pi_{i+\lambda}(-y)$ convex is a sufficient condition for this to hold), that $L_i(y) < \infty$ for all y, that $(c_{i-1} - \alpha c_i)y + L_{i-1}(y) \to \infty$ as $|y| \to \infty$, that $E(D_i) < \infty$, and finally that $t(x)$ is now given by

$$t(x) = -c_{n-\lambda+1}x.$$

For $i = 1, 2, \ldots, n - \lambda$, define $g_i(y)$ by

$$g_i(y) = c_i y + L_i(y) + \alpha \int_0^\infty f_{i+1}(y - z)\phi_i(z)\, dz. \tag{12.12}$$

Then (12.10) reduces to

$$f_i(x) = \min_{y > x} \left[K_i \delta(y - z) + g_i(y) \right] - c_i x.$$

Let S_i be the smallest value of y that minimizes $g_i(y)$, and let s_i be the smallest value of y less than or equal to S_i such that $g_i(s_i) = g_i(S_i) + K_i$. (We shall see below that S_i and s_i are well defined.) The situation is depicted in Figure 12.4.

An (s_i, S_i) *policy* is such that if $x_i \geq s_i$, then no order is placed ($y_i = x_i$); and if $x_i < s_i$, then we order up to S_i ($y_i = S_i$). We shall now show that under the above reasonably general conditions, there is an optimal policy in period i that is an (s_i, S_i) policy. Since the proof is rather long, we break it up into four parts.

Lemma 12.6. If $g_i(y)$ is continuous, $g_i(y) \to \infty$ as $|y| \to \infty$, and $g_i(y)$ is K_i-convex, then the (s_i, S_i) policy is optimal in period i.
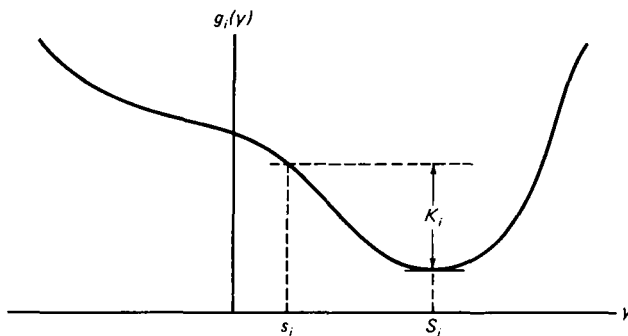
Figure 12.4

Proof: We first show that S_i and s_i are well defined. Let $w = g_i(0)$. Since $g_i(y) \to \infty$ as $|y| \to \infty$, there exists a y^1 such that $g_i(y) > w$ for $|y| > y^1$. Therefore, if $g_i(y)$ has a minimum, it must lie in the interval $[-y^1, y^1]$. However, it is well known that a function continuous on a closed interval has a minimum there. This fact and the continuity of $g_i(y)$ imply that S_i is well defined.

Let y^2 be such that $y^2 < S_i$ and $g_i(y) > g_i(S_i) + K_i$ for $y \leqslant y^2$. Consider the interval $[y^2, S_i]$. It is also well known that a function continuous on an interval assumes a value every number between any two of its values. It follows that s_i is well defined.

We shall now show that the (s_i, S_i) policy is optimal. There are two cases to consider.

Case 1: $x < s_i$. The cost of not ordering is $g_i(x) - c_i x$. By the definition of s_i, the continuity of $g_i(y)$, and the fact that $g_i(y) \to \infty$ as $y \to -\infty$, $g_i(x) > g_i(S_i) + K_i$. Thus, $g_i(x) - c_i x > g_i(S_i) + K_i - c_i x$ which is the cost of ordering up to S_i. Since S_i minimizes $g_i(y)$, it follows that $y = S_i$ is optimal for $x < s_i$.

Case 2: $s_i \leqslant x$. If $x = s_i$, then $g_i(x) = g_i(S_i) + K_i$. It follows that $y = x$ is optimal. Consider $s_i < x$. If it pays to order up to some $y > x$, then $g_i(x) > g_i(y) + K_i \geqslant g_i(S_i) + K_i = g_i(s_i)$. Let $a = y - x > 0$ and $b = x - s_i > 0$. It follows that

$$g_i(x) + \frac{a}{b} \left[g_i(x) - g_i(s_i) \right] > g_i(y) + K_i.$$

However, this implies that

$$K_i + g_i(y) - g_i(x) - \frac{a}{b} \left[g_i(x) - g_i(s_i) \right] < 0,$$

which contradicts the K_i-convexity of $g_i(y)$. Therefore, $y = x$ is optimal for $s_i < x$. ∎

Lemma 12.7. If the (s_i, S_i) policy is optimal in period i and $g_i(y)$ is continuous, then $g_{i-1}(y)$ is continuous and $g_{i-1}(y) \to \infty$ as $|y| \to \infty$.

Proof: Since $c_{i-1}y$ and $L_{i-1}(y)$ are each continuous (a convex function is continuous), it follows from (12.12) that $g_{i-1}(y)$ is continuous if we can show that $\int_0^\infty f_i(y-z)\phi_{i-1}(z)\, dz$ is continuous. It follows from the optimality of the (s_i, S_i) policy in period i that $f_i(x)$ is given by

$$f_i(x) = \begin{cases} g_i(S_i) + K_i - c_i x & \text{if } x < s_i, \\ g_i(x) - c_i x & \text{if } x \geqslant s_i. \end{cases} \tag{12.13}$$

It is clear from (12.13), the continuity of $g_i(x)$, and the definition of s_i that $f_i(x)$ is continuous for all x.

Problem 12.6. Prove that $\int_0^\infty f_i(y-z)\phi_{i-1}(z)\, dz$ is continuous and, thus, that $g_{i-1}(y)$ is also.

We now show that $g_{i-1}(y) \to \infty$ as $|y| \to \infty$. It follows from (12.12) and (12.13) that for $y > s_i$,

$$g_{i-1}(y) = (c_{i-1} - \alpha c_i)y + L_{i-1}(y) + \alpha \int_0^{y-s_i} g_i(y-z)\phi_{i-1}(z)\, dz$$

$$+ \alpha \big[g_i(S_i) + K_i \big] \int_{y-s_i}^\infty \phi_{i-1}(z)\, dz + \alpha c_i E(D_{i-1})$$

$$\geqslant (c_{i-1} - \alpha c_i)y + L_{i-1}(y)$$

$$+ \alpha \bigg[g_i(S_i) + K_i \int_{y-s_i}^\infty \phi_{i-1}(z)\, dz + c_i E(D_{i-1}) \bigg].$$

Therefore, by our assumptions at the beginning of this section, $g_{i-1}(y) \to \infty$ as $y \to \infty$. In a similar fashion it can be shown that for $y < s_i$,

$$g_{i-1}(y) = (c_{i-1} - \alpha c_i)y + L_{i-1}(y) + \alpha \big[g_i(S_i) + K_i + c_i E(D_{i-1}) \big].$$

Therefore, $g_{i-1}(y) \to \infty$ as $y \to -\infty$. ∎

Lemma 12.8. If the (s_i, S_i) policy is optimal in period i and $g_i(y)$ is K_i-convex, then $g_{i-1}(y)$ is K_{i-1}-convex.

Proof: By (12.12), Problem 12.3, and $K_{i-1} \geqslant \alpha K_i$, it suffices to show that $f_i(x)$ is K_i-convex. It follows from the optimality of the (s_i, S_i) policy in period i that $f_i(x)$ is given by (12.13). To show that $f_i(x)$ is K_i-convex, pick an $x \in E^1$ and $a, b > 0$. There are four cases to consider, depending on the relation of s_i to $x - b$, x, and $x + a$.

Case 1: $x - b \geqslant s_i$. In this interval

$$K_i + f_i(x+a) - f_i(x) - \frac{a}{b}\big[f_i(x) - f_i(x-b) \big]$$

$$= K_i + g_i(x+a) - g_i(x) - \frac{a}{b}\big[g_i(x) - g_i(x-b) \big] \geqslant 0.$$

The last inequality holds since $g_i(y)$ is K_i-convex.

Case 2: $x - b < s_i \leqslant x$. In this interval

$$K_i + f_i(x + a) - f_i(x) - \frac{a}{b}\left[f_i(x) - f_i(x - b)\right]$$

$$= K_i + g_i(x + a) - g_i(x) - \frac{a}{b}\left[g_i(x) - g_i(S_i) - K_i\right]$$

$$= K_i + g_i(x + a) - g_i(x) - \frac{a}{b}\left[g_i(x) - g_i(s_i)\right] \equiv V.$$

Note that $K_i + g_i(x + a) - g_i(x) \geqslant 0$ since otherwise it would pay to order from x to $x + a$, which is excluded by the optimality of the (s_i, S_i) policy. If $x = s_i$, then $V = K_i + g_i(x + a) - g_i(x) \geqslant 0$. Therefore, assume that $x > s_i$. If $g_i(x) - g_i(s_i) \leqslant 0$, then clearly $V \geqslant 0$. Therefore, suppose that $g_i(x) - g_i(s_i) > 0$. This can only be true for $x > S_i$. Define V' by

$$V' \equiv K_i + g_i(x + a) - g_i(x) - \frac{a}{x - s_i}\left[g_i(x) - g_i(s_i)\right].$$

Since $g_i(y)$ is K_i-convex, $V' \geqslant 0$. Furthermore, since $b > x - s_i$, $V \geqslant V'$. Therefore, $V \geqslant 0$ in all cases and $f_i(x)$ is K_i-convex in this interval.

Case 3: $x < s_i \leqslant x + a$. In this interval

$$K_i + f_i(x + a) - f_i(x) - \frac{a}{b}\left[f_i(x) - f_i(x - b)\right] = g_i(x + a) - g_i(S_i) \geqslant 0.$$

The last inequality holds since S_i minimizes $g_i(y)$.

Case 4: $x + a \leqslant s_i$. In this interval

$$K_i + f_i(x + a) - f_i(x) - \frac{a}{b}\left[f_i(x) - f_i(x - b)\right] = K_i \geqslant 0. \quad \blacksquare$$

Theorem 12.9. The (s_i, S_i) policy is optimal in period i for all i, $1 \leqslant i \leqslant n - \lambda$.

Proof: We first show by induction that $g_i(y)$ is continuous, that $g_i(y) \to \infty$ as $|y| \to \infty$, and that $g_i(y)$ is K_i-convex for all i, $1 \leqslant i \leqslant n - \lambda$. It is easy to show that $g_{n-\lambda}(y)$ is given by

$$g_{n-\lambda}(y) = (c_{n-\lambda} - \alpha c_{n-\lambda+1})y + L_{n-\lambda}(y) + \alpha c_{n-\lambda+1}E\left(\sum_{i=n-\lambda}^{n} D_i\right).$$

$$(12.14)$$

It follows from (12.14) that $g_{n-\lambda}(y)$ has the desired properties. Assume now that $g_i(y)$ has the desired properties for some i, $1 < i \leqslant n - \lambda$. We must show the same for $g_{i-1}(y)$. It follows from Lemmas 12.6 and 12.7 that $g_{i-1}(y)$ is continuous and $g_{i-1}(y) \to \infty$ as $|y| \to \infty$. Furthermore, it follows from Lemmas 12.6 and 12.8 that $g_{i-1}(y)$ is K_{i-1}-convex. Therefore, $g_i(y)$ has the desired properties for all i and it follows from Lemma 12.6 that the (s_i, S_i) policy is optimal in period i for all i. $\quad \blacksquare$

The concept of K_i-convexity and the first proof of Lemma 12.8 were given in Scarf [3]. However, our proof, which did not assume that $g_i(y)$ is differentiable, is based on Zabel [7].

To compute an optimal ordering policy for period i $(i = n - \lambda$, ..., 1), we begin by computing $g_i(y)$ for all relevant values of y. Then S_i is the smallest value of y that minimizes $g_i(y)$, s_i is the smallest value of y, $y \leqslant S_i$, such that $g_i(s_i) = g_i(S_i) + K_i$, and $f_i(x)$ can be simply computed from (12.13). Thus, only a single minimization is required for each period. On the other hand, the more general model considered in Chapter 11, Section 3 (i.e., recurrence relation (11.11)) required that one minimization be performed for each value of x. Thus, knowing that the (s_i, S_i) policy is optimal in period i, results in a considerable computational saving.

In this section we have assumed that all variables are continuous. However, by appropriately modifying our definitions and proofs (e.g., define s_i to be the smallest value of y, $y \leqslant S_i$, such that $g_i(s_i) \leqslant g_i(S_i) + K_i$), it is possible to show that the (s_i, S_i) policy is optimal when all variables are integer valued.

5. Optimality of Single Critical Number Policies

Suppose we further specialize the inventory model of the previous section by assuming that the set-up cost K_i is equal to zero. Therefore, the cost of ordering z items in period i is given by

$$c_i(z) = c_i z \qquad \text{for} \quad i = 1, 2, \ldots, n - \lambda.$$

Since S_i was defined to be the smallest value of y that minimizes $g_i(y)$ and s_i was defined to be the smallest value of y, $y \leqslant S_i$, for which $g_i(s_i) = g_i(S_i) + K_i$, it follows that $s_i = S_i$ for all i. A *single critical number policy* with critical number $\bar{y}_i$ is such that if $x_i \geqslant \bar{y}_i$, then no order is placed ($y_i = x_i$); and if $x_i < \bar{y}_i$, then we order up to $\bar{y}_i$ ($y_i = \bar{y}_i$). Let $\bar{y}_i$ denote the common value of s_i and S_i. It follows from Theorem 12.9 that the single critical number policy with critical number $\bar{y}_i$ is optimal for period i.

Let us now assume that the cost functions and the demand distribution are stationary. That is $c_i = c$, $L_i(y) = L(y)$, and $\phi_i(z) = \phi(z)$ for all i. We shall now show that there is one critical number that is optimal for all values of i.

Theorem 12.10. If the cost functions and the demand distribution are stationary, then there is a single critical number policy that is optimal for period i and, furthermore, $\bar{y}_i = \bar{y}$ for all i.

Proof: We shall use induction on i. From (12.14) we get

$$g_{n-\lambda}(y) = c(1 - \alpha)y + L(y) + \alpha(\lambda + 1)cE(D). \qquad (12.15)$$

Let $\bar{y}$ be the smallest value of y that minimizes $g_{n-\lambda}(y)$. We shall show that $\bar{y}$ is the smallest value of y that minimizes $g_i(y)$ for all i. Suppose that this is true for some i, $1 < i \leqslant n - \lambda$. We must show the same for $i - 1$. It

follows from (12.13) that $f_i(x)$ is given by

$$f_i(x) = \begin{cases} g_i(\bar{y}) - cx & \text{if } x < \bar{y}, \\ g_i(x) - cx & \text{if } x \geqslant \bar{y}. \end{cases} \tag{12.16}$$

Furthermore, it follows from (12.12), (12.15), (12.16), and the definition of $\bar{y}$ that for $y < \bar{y}$,

$$g_{i-1}(y) = c(1 - \alpha)y + L(y) + \alpha c E(D) + \alpha g_i(\bar{y})$$
$$> c(1 - \alpha)\bar{y} + L(\bar{y}) + \alpha c E(D) + \alpha g_i(\bar{y}) = g_{i-1}(\bar{y}).$$

Similarly, for $y > \bar{y}$,

$$g_{i-1}(y) = c(1 - \alpha)y + L(y) + \alpha c E(D)$$
$$+ \alpha \left[\int_0^{y-\bar{y}} g_i(y - z)\phi(z) \, dz + g_i(\bar{y}) \int_{y-\bar{y}}^{\infty} \phi(z) \, dz \right]$$
$$\geqslant c(1 - \alpha)\bar{y} + L(\bar{y}) + \alpha c E(D) + \alpha g_i(\bar{y}) = g_{i-1}(\bar{y}).$$

It follows that $\bar{y}$ is the smallest value of y that minimizes $g_{i-1}(y)$ and therefore the proof is complete. ∎

Thus, in the stationary case, an optimal ordering policy can be specified for all periods by just one critical number. For additional inventory models for which a single critical number policy is optimal, see the interesting paper by Veinott [5].

References

[1] Eppen, G. D., F. J. Gould, and B. P. Pashigian, "Extensions of the Planning Horizon Theorem in the Dynamic Lot Size Model," *Management Sci.* 15 (1969), 268–277.

[2] Luenberger, D. G., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Massachusetts, 1973.

[3] Scarf, H., "The Optimality of (S, s) Policies in the Dynamic Inventory Problem," *Mathematical Methods in the Social Sciences*, Stanford Univ. Press, Stanford, California (1960).

[4] Topkis, D. M., Class notes for IEOR 264, Department of Industrial Engineering and Operations Research, Univ. California, Berkeley (1972).

[5] Veinott, A. F., Jr., "The Optimal Inventory Policy for Batch Ordering," *Operations Res.* 13 (1965), 424–432.

[6] Wagner, H. M. and T. M. Whitin, "Dynamic Version of the Economic Lot Size Model," *Management Sci.* 5 (1958), 89–96.

[7] Zabel, E., "A Note on the Optimality of (S, s) Policies in Inventory Theory," *Management Sci.* 9 (1962), 123–125.

[8] Zangwill, W. I., "A Deterministic Multi-Period Production Scheduling Model with Backlogging," *Management Sci.* 13 (1966), 105–119.

[9] Zangwill, W. I., "A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System—A Network Approach," *Management Sci.* 15 (1969), 506–527.

Chapter 13

MARKOVIAN DECISION PROCESSES

1. Introduction

In this chapter we present a general framework for describing and solving certain probabilistic dynamic-programming problems. Our presentation, especially at the beginning of Section 2, will parallel the excellent treatment of Ross [3]. We shall also follow most of the notation developed there.

Suppose that at each of the times 0, 1, . . . a system is observed and determined to be in one of a countable number of states. We shall denote the set of all possible states by $I = \{0, 1, \ldots \}$. After determining the state of the system, one of a finite number of actions must be taken (decisions must be made). We shall denote the set of all possible actions by K. However, in general, the number of possible actions may depend on the state of the system. (See the example in Section 3.)

Let X_t and A_t be, respectively, the state of the system and subsequent action at time t. If $X_t = i$ and $A_t = a$, then we shall denote the probability that the system will be in state j at time $t + 1$ by $P_{ij}(a)$. This notation implies that the $P_{ij}(a)$ depend on neither t nor the entire history of the system prior to time t, $H_t = \{X_0, A_0, X_1, A_1, \ldots, X_{t-1}, A_{t-1}\}$. That is,

$$P\{X_{t+1} = j \mid H_t, X_t = i, A_t = a\} = P\{X_{t+1} = j \mid X_t = i, A_t = a\} = P_{ij}(a).$$
(13.1)

In addition, if $X_t = i$ and $A_t = a$, then an expected cost $C(i, a)$ is incurred. It is assumed that the $C(i, a)$ are bounded; i.e., there exists a number M such that $|C(i, a)| < M$ for all $i \in I$, $a \in K$.

A *policy*, which we will denote by Π, is a rule for choosing an action at each point in time. In general, the action specified by a policy at time t may depend not only on the present state X_t, but also possibly on the history H_t and on t itself. A policy may also be randomized in that it

chooses each of the possible actions with a certain probability. However, for most of the problems that we consider, it will be sufficient to restrict our attention to the set of stationary policies. A policy is *stationary* if the action it specifies at time t depends only on the present state X_t and if it is nonrandomized.

Given X_0 and a policy Π, the sequence $\{X_t, A_t, t = 0, 1, \ldots \}$ is a well-defined stochastic process which we call a *Markovian decision process*. The term Markovian is used because of our assumption concerning the transition probabilities (i.e., the assumption given by (13.1)).

As an example of a Markovian decision process, consider the inventory model with lost sales and zero delivery lag of Chapter 11, Section 2. (Assume that the number of different order quantities is finite and that costs are bounded.) Let X_t be the inventory level at the beginning of period t and let A_t be the number of items ordered at this time. If $X_t = i$ and $A_t = a$, then we incur an expected cost $C(i, a)$ which is given by

$$C(i, a) = c(a) + L^0(i + a),$$

where $c(a)$ is the cost of ordering a items and $L^0(i + a)$ is the expected holding and shortage cost given $i + a$ items to meet demand. In addition, the next state is chosen in accordance with the transition probabilities $P_{ij}(a)$, which are given by

$$P_{ij}(a) = \begin{bmatrix} P\{D = i + a - j\} \equiv p(i + a - j) & \text{if } j > 0, \\ \sum_{d = i + a}^{\infty} P\{D = d\} \equiv \sum_{d = i + a}^{\infty} p(d) & \text{if } j = 0, \end{bmatrix}$$

where D is the stationary demand per period.

In order to evaluate two or more policies, we must have some criterion for comparison. Two general problems are of interest, depending on whether the process evolves for a finite or an infinite number of periods. Let us begin with the finite-period (horizon) problem. For any policy Π and discount factor $0 < \alpha \leqslant 1$, let $V_{\Pi, \alpha}(i, n)$ be the expected total discounted cost for periods 0 through $n - 1$ given that $X_0 = i$ and the policy Π is used. Then $V_{\Pi, \alpha}(i, n)$ is given by

$$V_{\Pi, \alpha}(i, n) = E_{\Pi}\left[\sum_{t=0}^{n-1} \alpha^t C(X_t, A_t) | X_0 = i \right] \qquad (i \in I), \qquad (13.2)$$

where E_{Π} is the conditional expectation given that the policy Π is used. (Note that $E_{\Pi}[C(X_t, A_t)|X_0 = i]$ is well defined since costs are bounded; hence (13.2) is also well defined.)

In the finite-horizon problem, the objective is to find a policy Π^* whose corresponding expected discounted cost $V_{\Pi^*, \alpha}(i, n)$ is minimal. As you might suspect, this problem can be solved by standard dynamic-

programming techniques. Define the optimal expected value function $V_\alpha(i, k)$ as follows:

$V_\alpha(i, k) =$ the minimum expected discounted cost (at
discount rate α) for k periods given that the
process starts in state i.   (13.3)

Then it is clear that $V_\alpha(i, k)$ satisfies the recurrence relation

$$V_\alpha(i, k) = \min_a \left[C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)V_\alpha(j, k-1) \right] \quad (k = 1, 2, \ldots, n)$$

(13.4)

and the appropriate boundary condition is

$$V_\alpha(i, 0) = 0 \quad \text{for} \quad i \in I. \tag{13.5}$$

(For a proof of (13.4), which allows for the possibility of randomized policies, see Derman [1].) If $X_0 = i$, then the answer to our problem is, of course, $V_\alpha(i, n)$.

Up to this point in the book, we have always assumed that the process of interest evolves for only a finite number of periods. However, sometimes one wishes to assume that a system will be in operation indefinitely. For such an infinite-period (horizon) problem and any policy Π, let $V_{\Pi, \alpha}(i)$ be the expected total discounted cost for periods 0, 1, ... given that $X_0 = i$ and the policy Π is used. Then $V_{\Pi, \alpha}(i)$ is given by

$$V_{\Pi, \alpha}(i) = E_\Pi \left[\sum_{t=0}^{\infty} \alpha^t C(X_t, A_t) | X_0 = i \right] \quad (i \in I), \tag{13.6}$$

where now $0 < \alpha < 1$. (Note that (13.6) is well defined since costs are bounded and $\alpha < 1$.)

Let

$$V_\alpha(i) = \inf_\Pi V_{\Pi, \alpha}(i), \quad i \in I.$$

(The infimum of a set S, abbreviated inf S, is equal to the largest number y such that $y \leqslant s$ for every $s \in S$. If S has a minimum, then the infimum is equal to it.) In the infinite-horizon problem, our objective will be to find a policy Π^* such that

$$V_{\Pi^*, \alpha}(i) = V_\alpha(i) \quad \text{for all} \quad i \in I.$$

We shall call such a policy α-*optimal*.

Since there are an infinite number of policies Π, it is not clear that an α-optimal policy exists. This will be the subject of Section 2. Furthermore, even if an α-optimal policy exists, there is the problem of how to find it. Standard dynamic programming cannot be used since there are an infinite number of periods. In Section 3 we shall present some computational procedures that can be used to find an α-optimal policy.

Before concluding this section, let us mention another interesting criterion for the infinite-horizon problem. Let $\phi_\Pi(i)$ be the long-run expected average cost per period given that $X_0 = i$ and the policy Π is used. Then $\phi_\Pi(i)$ is given by

$$\phi_\Pi(i) = \lim_{n \to \infty} \frac{E_\Pi\left[\sum_{t=0}^{n-1} C(X_t, A_t) | X_0 = i\right]}{n} \qquad (i \in I), \qquad (13.7)$$

assuming the limit on the r.h.s. of (13.7) exists. Another possible objective is to find a policy Π^* such that

$$\phi_{\Pi^*}(i) = \inf_\Pi \phi_\Pi(i) \qquad \text{for all} \quad i \in I.$$

Such a policy is said to be *average cost optimal*. We shall not discuss the average cost case in this book since it has many technical difficulties that the discounted case does not possess. For discussions of the average cost case, see Derman [1], Howard [2], and Ross [3].

2. Existence of an Optimal Policy

In this section we show that an α-optimal policy exists and, in fact, it can be taken to be stationary.

We begin with a theorem that gives a functional equation satisfied by the optimal expected value function V_α. It is called a functional equation rather than a recurrence relation since the same function appears on both sides of the equation.

Theorem 13.1.

$$V_\alpha(i) = \min_a \left[C(i, a) + \alpha \sum_{j=0}^\infty P_{ij}(a) V_\alpha(j) \right] \qquad (i \in I). \qquad (13.8)$$

Proof: We shall give an intuitive justification. (See Ross [3] for a mathematically rigorous proof.) If $X_0 = i$ and $A_0 = a$, then we incur an immediate expected cost $C(i, a)$. Since there are still an infinite number of periods remaining at the beginning of time period 1, the minimum expected discounted cost from time 1 on is $V_\alpha(j)$ when $X_1 = j$. Furthermore, since $X_1 = j$ with probability $P_{ij}(a)$ and we must multiply $V_\alpha(j)$ by α to discount to time 0, it follows that $C(i, a) + \alpha \sum_{j=0}^\infty P_{ij}(a) V_\alpha(j)$ is the minimum expected discounted cost when $X_0 = i$ and $A_0 = a$. Let a_i be the action (or an action) whose corresponding (minimum) expected discounted cost is minimal. Then it is clearly optimal to choose action a_i at time 0 with probability one. The correctness of (13.8) now follows. ∎

Recall that the action specified by a stationary policy depends only on the present state. Thus, a stationary policy f can be thought of as a mapping from the set of states I into the set of actions K. When the process is in state i, the stationary policy f chooses action $f(i)$.

Let $B(I)$ be the set of bounded real-valued functions on the set I. Since costs are bounded, it follows that $V_{\Pi, \alpha} \in B(I)$ for all policies Π. For any stationary policy f, define the mapping

$$T_f: B(I) \to B(I)$$

as follows:

$$(T_f u)(i) = C[i, f(i)] + \alpha \sum_{j=0}^{\infty} P_{ij}[f(i)]u(j) \qquad \text{for} \quad u \in B(I). \quad (13.9)$$

Thus, $T_f u$ is that member of $B(I)$ whose value at i is given by (13.9).

If $u, v \in B(I)$, then we shall write $u = v$ ($u \leqslant v$) to mean that $u(i) = v(i)$ ($u(i) \leqslant v(i)$) for all $i \in I$. Let $T_f^1 u = T_f u$ and let $T_f^n u = T_f(T_f^{n-1}u)$ for $n > 1$. The following lemma gives some properties of the mapping T_f which we shall find useful.

Lemma 13.2. The mapping T_f has the following properties:

(i) $u \leqslant v$ implies $T_f u \leqslant T_f v$,
(ii) $T_f V_{f, \alpha} = V_{f, \alpha}$,
(iii) $\lim_{n \to \infty} T_f^n u = V_{f, \alpha}$ \qquad for all $u \in B(I)$.

Proof: Part (i) is obvious from the definition of T_f. Part (ii) is equivalent to

$$V_{f, \alpha}(i) = C[i, f(i)] + \alpha \sum_{j=0}^{\infty} P_{ij}[f(i)]V_{f, \alpha}(j),$$

which can be shown to be true in a manner similar to the way we proved Theorem 13.1. To prove part (iii), let q_{ik}^n be the probability of going from state i to state k in n time periods when using stationary policy f. The probabilities q_{ik}^n can be recursively computed as

$$q_{ik}^1 \equiv P_{ik}[f(i)],$$

$$q_{ik}^n = \sum_{j=0}^{\infty} P_{ij}[f(i)]q_{jk}^{n-1} \qquad \text{for} \quad n > 1.$$

We shall first show by induction that

$$(T_f^n u)(i) = V_{f, \alpha}(i, n) + \alpha^n \sum_{k=0}^{\infty} q_{ik}^n u(k) \qquad \text{for all} \quad n \geqslant 1,$$

where $V_{f, \alpha}(i, n)$ was defined by (13.2). The case $n = 1$ immediately follows from the definition of T_f. Therefore, assume that $T_f^{n-1}u$ has the desired

form and consider $T_f^n u$. We get

$$(T_f^n u)(i) = C[i, f(i)] + \alpha \sum_{j=0}^{\infty} P_{ij}[f(i)]\left[V_{f,\alpha}(j, n-1) + \alpha^{n-1} \sum_{k=0}^{\infty} q_{jk}^{n-1} u(k)\right]$$

$$= C[i, f(i)] + \alpha \sum_{j=0}^{\infty} P_{ij}[f(i)]V_{f,\alpha}(j, n-1)$$

$$+ \alpha^n \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} P_{ij}[f(i)] q_{jk}^{n-1} u(k)$$

$$= V_{f,\alpha}(i, n) + \alpha^n \sum_{k=0}^{\infty} q_{ik}^n u(k).$$

The desired result now follows by letting $n \to \infty$ since costs are bounded and $\alpha < 1$. (Note that the convergence in part (iii) is uniform on the set I.) ∎

The next theorem, which is the most important of this section, shows that there exists a stationary policy that is α-optimal and, furthermore, that it is determined by the functional equation (13.8).

Theorem 13.3. If f_α is the stationary policy that chooses in state i the action minimizing

$$C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)V_\alpha(j),$$

then f_α is α-optimal.

Proof: We shall show by induction that
$$T_{f_\alpha}^n V_\alpha = V_\alpha \qquad \text{for all} \quad n \geqslant 1.$$
If we apply the mapping T_{f_α} to V_α, we get

$$(T_{f_\alpha} V_\alpha)(i) = C[i, f_\alpha(i)] + \alpha \sum_{j=0}^{\infty} P_{ij}[f_\alpha(i)]V_\alpha(j)$$

$$= \min_a \left[C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)V_\alpha(j) \right]$$

$$= V_\alpha(i),$$

where the last equation follows from Theorem 13.1. Therefore, $T_{f_\alpha}^1 V_\alpha = V_\alpha$. Assume that $T_{f_\alpha}^{n-1} V_\alpha = V_\alpha$. Then

$$T_{f_\alpha}^n V_\alpha = T_{f_\alpha}(T_{f_\alpha}^{n-1} V_\alpha) = T_{f_\alpha} V_\alpha = V_\alpha.$$

Letting $n \to \infty$ and using part (iii) of Lemma 13.2 gives $V_{f_\alpha, \alpha} = V_\alpha$, which is the desired result. ∎

Before considering procedures that can actually be used to determine

an α-optimal policy, we need some additional results. Define the mapping

$$T_\alpha: B(I) \to B(I)$$

as

$$(T_\alpha u)(i) = \min_a \left[C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)u(j) \right] \qquad \text{for} \quad u \in B(I). \quad (13.10)$$

Lemma 13.4. $\lim_{n \to \infty} T_\alpha^n u = V_\alpha$ for all $u \in B(I)$.

Proof: We shall show by induction that there exists a number B such that

$$|(T_\alpha^n u)(i) - V_\alpha(i)| < \alpha^n B \quad (i \in I) \qquad \text{for all} \quad n \geqslant 1.$$

Since $u \in B(I)$, there exists a number B' such that $|u(i)| < B'$ for all $i \in I$. Furthermore, it is easy to show that $|V_\alpha(i)| < M/(1 - \alpha)$ for all $i \in I$. Let $B = 2 \max[B', M/(1 - \alpha)]$. Also let $\bar{a}_i$ be the action that minimizes

$$C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)V_\alpha(j)$$

and let a_i^1 be the action that minimizes

$$C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a)u(j).$$

Then

$$(T_\alpha u)(i) - V_\alpha(i) = \left[C(i, a_i^1) + \alpha \sum_{j=0}^{\infty} P_{ij}(a_i^1)u(j) \right]$$

$$- \left[C(i, \bar{a}_i) + \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i)V_\alpha(j) \right]$$

$$\leqslant \left[C(i, \bar{a}_i) + \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i)u(j) \right]$$

$$- \left[C(i, \bar{a}_i) + \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i)V_\alpha(j) \right]$$

$$= \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i)[u(j) - V_\alpha(j)]$$

$$< \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i)B = \alpha B$$

and

$$V_\alpha(i) - (T_\alpha u)(i) \leqslant \alpha \sum_{j=0}^{\infty} P_{ij}(a_i^1)[V_\alpha(j) - u(j)] < \alpha \sum_{j=0}^{\infty} P_{ij}(a_i^1)B = \alpha B.$$

Therefore, $|(T_\alpha u)(i) - V_\alpha(i)| < \alpha B$. Assume that the desired result is true for $n - 1$ and let a_i^n be the action that minimizes

$$C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a) (T_\alpha^{n-1} u)(j).$$

Then

$$(T_\alpha^n u)(i) - V_\alpha(i) \leqslant \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i) \left[(T_\alpha^{n-1} u)(j) - V_\alpha(j) \right]$$

$$< \alpha \sum_{j=0}^{\infty} P_{ij}(\bar{a}_i) \alpha^{n-1} B = \alpha^n B$$

and

$$V_\alpha(i) - (T_\alpha^n u)(i) \leqslant \alpha \sum_{j=0}^{\infty} P_{ij}(a_i^n) \left[V_\alpha(j) - (T_\alpha^{n-1} u)(j) \right]$$

$$< \alpha \sum_{j=0}^{\infty} P_{ij}(a_i^n) \alpha^{n-1} B = \alpha^n B.$$

Therefore, $|(T_\alpha^n u)(i) - V_\alpha(i)| < \alpha^n B$ for all $n \geqslant 1$ and the lemma is proven. ∎

The idea of applying T_α to an arbitrary function $u \in B(I)$, then applying T_α to the function that results, etc. is known as the *method of successive approximations* and will be discussed in greater detail in the next section.

Theorem 13.5. V_α is the unique solution to Eq. (13.8).

Proof: Suppose that there exists another function V such that

$$V(i) = \min_a \left[C(i, a) + \alpha \sum_{j=0}^{\infty} P_{ij}(a) V(j) \right] \qquad (i \in I).$$

Using a proof almost identical to the one used to prove Theorem 13.3, it can be shown that

$$T_\alpha^n V = V \qquad \text{for all} \quad n \geqslant 1.$$

Letting $n \to \infty$ and using Lemma 13.4 gives $V = V_\alpha$, which is the desired result. ∎

3. Computational Procedures

In this section the set of possible states is assumed to be finite and will be denoted by $I_m = \{0, 1, \ldots, m\}$.

The first procedure that we consider is called the *policy improvement algorithm.* It is due to Howard [2].

Problem 13.1. Let f be a stationary policy. Show that $V_{f,\alpha}$ is the unique solution to the system of equations

$$V_{f,\alpha}(i) = C[i, f(i)] + \alpha \sum_{j=0}^{m} P_{ij}[f(i)]V_{f,\alpha}(j) \qquad (i = 0, 1, \ldots, m). \quad (13.11)$$

Thus we may solve this system of $m + 1$ linear equations for the $m + 1$ unknown values of $V_{f,\alpha}(i)$. Now let g be the stationary policy that chooses in state i the action (or an action) minimizing

$$C(i, a) + \alpha \sum_{j=0}^{m} P_{ij}(a)V_{f,\alpha}(j). \quad (13.12)$$

However, we shall take $g(i)$ different from $f(i)$ only if $g(i)$ results in a strict improvement over $f(i)$ in (13.12). The following theorem shows that g is at least as good as f.

Theorem 13.6. If f and g are stationary policies with g defined as above, then

  (i) $V_{g,\alpha}(i) \leqslant V_{f,\alpha}(i)$ for all $i \in I_m$,
  (ii) $V_{g,\alpha}(i) < V_{f,\alpha}(i)$ for each i for which $g(i)$ results in a strict improvement over $f(i)$ in (13.12).

Proof: (i)

$$(T_g V_{f,\alpha})(i) = C[i, g(i)] + \alpha \sum_{j=0}^{m} P_{ij}[g(i)]V_{f,\alpha}(j)$$

$$\leqslant C[i, f(i)] + \alpha \sum_{j=0}^{m} P_{ij}[f(i)]V_{f,\alpha}(j) = V_{f,\alpha}(i).$$

Therefore, $T_g V_{f,\alpha} \leqslant V_{f,\alpha}$. Assume that $T_g^{n-1}V_{f,\alpha} \leqslant T_g V_{f,\alpha} \leqslant V_{f,\alpha}$ and consider $T_g^n V_{f,\alpha}$. We get

$$T_g^n V_{f,\alpha} = T_g(T_g^{n-1}V_{f,\alpha}) \leqslant T_g V_{f,\alpha} \leqslant V_{f,\alpha}. \quad (13.13)$$

Letting $n \to \infty$ and using part (iii) of Lemma 13.2 gives $V_{g,\alpha} \leqslant V_{f,\alpha}$.

  (ii) If $g(i)$ results in a strict improvement over $f(i)$ in (13.12), then (13.13) may be replaced by

$$(T_g^n V_{f,\alpha})(i) \leqslant (T_g V_{f,\alpha})(i) < V_{f,\alpha}(i).$$

Letting $n \to \infty$ gives $V_{g,\alpha}(i) < V_{f,\alpha}(i)$. ∎

Theorem 13.7. If $f(i) = g(i)$ for all $i \in I_m$, then f is α-optimal.

Proof: If $f(i) = g(i)$ for all $i \in I_m$, then $V_{f,\alpha}(i) = V_{g,\alpha}(i)$ for all $i \in I_m$. It then follows that

$$\min_a \left[C(i, a) + \alpha \sum_{j=0}^m P_{ij}(a) V_{f,\alpha}(j) \right] = C[i, g(i)] + \alpha \sum_{j=0}^m P_{ij}[g(i)] V_{f,\alpha}(j)$$

$$= C[i, g(i)] + \alpha \sum_{j=0}^m P_{ij}[g(i)] V_{g,\alpha}(j)$$

$$= V_{g,\alpha}(i) = V_{f,\alpha}(i).$$

Therefore, $V_{f,\alpha}$ satisfies Eq. (13.8) and, hence, by uniqueness (Theorem 13.5) $V_{f,\alpha} = V_\alpha$. ∎

We now formally state the above procedure as an algorithm. The policy improvement algorithm is as follows:

Step 1: For a stationary policy f, solve the system of $m + 1$ equations given by (13.11) for the $m + 1$ unknown values of $V_{f,\alpha}(i)$.

Step 2: Using the values of $V_{f,\alpha}(i)$ determined in step 1, find that stationary policy g such that, for each state i, $g(i)$ is the action that minimizes (13.12) ($g(i)$ is chosen different from $f(i)$ only if it results in a strict improvement over $f(i)$).

Step 3: If $f(i) \neq g(i)$ for at least one i, then go to step 1 and use g in place of f. If $f(i) = g(i)$ for all $i \in I_m$, then stop—f is α-optimal.

Corollary 13.8. The policy improvement algorithm converges to an optimal policy in a finite number of iterations (steps 1–3).

Proof: There are only a finite number of stationary policies. By Theorem 13.6 each iteration results in a strict improvement. Therefore, no policy will be repeated. When no further improvement is possible, then f will equal g and f will be α-optimal by Theorem 13.7. ∎

In order to use the policy improvement algorithm, some initial policy must be chosen. If there is no a priori basis for choosing a close-to-optimal policy, then it is often convenient to choose as an initial policy the one that minimizes the immediate expected costs. This is equivalent to starting at step 2 with $V_{f,\alpha}(i) = 0$ for all $i \in I_m$.

As an example of the policy improvement algorithm, consider the inventory model of Section 1 with $\alpha = 0.90$. However, to keep the state space finite, assume that the maximum inventory level is 3. Thus, the possible states (inventory levels) are 0, 1, 2, 3 and when the state is i, the possible actions (order quantities) are $0, \ldots, 3 - i$. Furthermore, the costs

Table 13.1. *Data for the example problem*

State i	Action a	$C(i, a)$	$P_{i0}(a)$	$P_{i1}(a)$	$P_{i2}(a)$	$P_{i3}(a)$
0	0	19.500	1	0	0	0
	1	15.125	.875	.125	0	0
	2	10.000	.625	.250	.125	0
	3	11.375	.125	.500	.250	.125
1	0	9.125	.875	.125	0	0
	1	8.000	.625	.250	.125	0
	2	9.375	.125	.500	.250	.125
2	0	2.000	.625	.250	.125	0
	1	7.375	.125	.500	.250	.125
3	0	1.375	.125	.500	.250	.125

and demand distribution are as follows:

$$c(0) = 0, \quad c(1) = 6, \quad c(2) = 8, \quad c(3) = 10;$$
$$h(z) = z \quad \text{for} \quad z = 0, 1, 2, 3;$$
$$\Pi(z) = 12z \quad \text{for} \quad z = 0, 1, 2, 3;$$
$$p(0) = \tfrac{1}{8}, \quad p(1) = \tfrac{1}{4}, \quad p(2) = \tfrac{1}{2}, \quad p(3) = \tfrac{1}{8}.$$

From these data we get Table 13.1. The reader should verify the correctness of Table 13.1.

Let $f_1, f_2, \ldots$ denote the sequence of stationary policies generated by the policy improvement algorithm. As an initial policy, we choose the one that minimizes the immediate expected costs. From the above table we get

$$f_1(0) = 2, \quad f_1(1) = 1, \quad f_1(2) = 0, \quad f_1(3) = 0.$$

To find the expected total discounted costs corresponding to f_1, we solve the following system of equations (step 1):

$$V_{f_1}(0) = 10.000 + .9\big[.625V_{f_1}(0) + .25V_{f_1}(1) + .125V_{f_1}(2)\big]$$
$$V_{f_1}(1) = 8.000 + .9\big[.625V_{f_1}(0) + .25V_{f_1}(1) + .125V_{f_1}(2)\big]$$
$$V_{f_1}(2) = 2.000 + .9\big[.625V_{f_1}(0) + .25V_{f_1}(1) + .125V_{f_1}(2)\big]$$
$$V_{f_1}(3) = 1.375 + .9\big[.125V_{f_1}(0) + .50V_{f_1}(1) + .250V_{f_1}(2) + .125V_{f_1}(3)\big].$$

(We neglect, here and below, the subscript .9 on values of $V_{f, .9}$.) The solution to these equations is

$$V_{f_1}(0) = 86.50, \quad V_{f_1}(1) = 84.50, \quad V_{f_1}(2) = 78.50, \quad V_{f_1}(3) = 75.26.$$

We now try to improve upon policy f_1. The calculations are as follows (step 2):

State 0

Action a $\qquad\qquad C(0, a) + .9 \sum_{j=0}^{3} P_{0j}(a) V_{f_1}(j)$

0	$19.500 + .9[\quad (86.50)$	$] = 97.35$
1	$15.125 + .9[.875(86.50) + .125(84.50)$	$] = 92.75$
2	$10.000 + .9[.625(86.50) + .250(84.50) + .125(78.50)$	$] = 86.50$
3	$11.375 + .9[.125(86.50) + .500(84.50) + .250(78.50) + .125(75.26)] = \boxed{85.26}$	

State 1

Action a $\qquad\qquad C(1, a) + .9 \sum_{j=0}^{3} P_{1j}(a) V_{f_1}(j)$

0	$9.125 + .9[.875(86.50) + .125(84.50)$	$] = 86.75$
1	$8.000 + .9[.625(86.50) + .250(84.50) + .125(78.50)$	$] = 84.50$
2	$9.375 + .9[.125(86.50) + .500(84.50) + .250(78.50) + .125(75.26)] = \boxed{83.26}$	

State 2

Action a $\qquad\qquad C(2, a) + .9 \sum_{j=0}^{3} P_{2j}(a) V_{f_1}(j)$

0	$2.000 + .9[.625(86.50) + .250(84.50) + .125(78.50)$	$] = \boxed{78.50}$
1	$7.375 + .9[.125(86.50) + .500(84.50) + .250(78.50) + .125(75.26)] = 81.26$	

State 3

Action a $\qquad\qquad C(3, a) + .9 \sum_{j=0}^{3} P_{3j}(a) V_{f_1}(j)$

0	$1.375 + .9[.125(86.50) + .500(84.50) + .250(78.50) + .125(75.26)] = \boxed{75.26}$

Therefore, the improved policy is

$$f_2(0) = 3, \qquad f_2(1) = 2, \qquad f_2(2) = 0, \qquad f_2(3) = 0.$$

Since f_2 is not equal to f_1, we return to step 1 and use f_2 in place of f_1. The system of equations corresponding to f_2 is as follows (step 1):

$$V_{f_2}(0) = 11.375 + .9\big[.125 V_{f_2}(0) + .500 V_{f_2}(1) + .250 V_{f_2}(2) + .125 V_{f_2}(3)\big]$$

$$V_{f_2}(1) = 9.375 + .9\big[.125 V_{f_2}(0) + .500 V_{f_2}(1) + .250 V_{f_2}(2) + .125 V_{f_2}(3)\big]$$

$$V_{f_2}(2) = 2.000 + .9\big[.625 V_{f_2}(0) + .250 V_{f_2}(1) + .125 V_{f_2}(2)\big]$$

$$V_{f_2}(3) = 1.375 + .9\big[.125 V_{f_2}(0) + .500 V_{f_2}(1) + .250 V_{f_2}(2) + .125 V_{f_2}(3)\big].$$

The solution to these equations is

$$V_{f_2}(0) = 77.73, \qquad V_{f_2}(1) = 75.73, \qquad V_{f_2}(2) = 70.71, \qquad V_{f_2}(3) = 67.73.$$

We now try to improve upon policy f_2. The calculations are as follows (step 2):

State 0

Action a
$$C(0, a) + .9 \sum_{j=0}^{3} P_{0j}(a)V_{f_2}(j)$$

0	$19.500 + .9[\quad(77.73)$	$] = 89.45$
1	$15.125 + .9[.875(77.73) + .125(75.73)$	$] = 84.85$
2	$10.000 + .9[.625(77.73) + .250(75.73) + .125(70.71)$	$] = 78.71$
3	$11.375 + .9[.125(77.73) + .500(75.73) + .250(70.71) + .125(67.73)] = \boxed{77.73}$	

State 1

Action a
$$C(1, a) + .9 \sum_{j=0}^{3} P_{1j}(a)V_{f_2}(j)$$

0	$9.125 + .9[.875(77.73) + .125(75.73)$	$] = 78.85$
1	$8.000 + .9[.625(77.73) + .250(75.73) + .125(70.71)$	$] = 76.71$
2	$9.375 + .9[.125(77.73) + .500(75.73) + .250(70.71) + .125(67.73)] = \boxed{75.73}$	

State 2

Action a
$$C(2, a) + .9 \sum_{j=0}^{3} P_{2j}(a)V_{f_2}(j)$$

0	$2.000 + .9[.625(77.73) + .250(75.73) + .125(70.71)$	$] = \boxed{70.71}$
1	$7.375 + .9[.125(77.73) + .500(75.73) + .250(70.71) + .125(67.73)] = 73.73$	

State 3

Action a
$$C(3, a) + .9 \sum_{j=0}^{3} P_{3j}(a)V_{f_2}(j)$$

0	$1.375 + .9[.125(77.73) + .500(75.73) + .250(70.71) + .125(67.73)] = \boxed{67.73}$

Therefore, the improved policy is
$$f_3(0) = 3, \qquad f_3(1) = 2, \qquad f_3(2) = 0, \qquad f_3(3) = 0.$$
Since f_3 is equal to f_2, it follows that f_2 is 0.90-optimal. Thus, although there are 24 different stationary policies, the policy improvement algorithm has found the optimal policy in just two iterations.

Let us now consider the method of successive approximations which was briefly discussed in Section 2. Recall that if u is an arbitrary function in $B(I)$, then
$$\lim_{n \to \infty} T_\alpha^n u = V_\alpha,$$
where the mapping T_α was defined by (13.10). (The same result is true for $u \in B(I_m)$.) Furthermore, if V_α has been determined, then the stationary policy that chooses in state i the action (or an action) minimizing the r.h.s. of Eq. (13.8) is α-optimal. In practice, $(T_\alpha^n u)(i)$ will be only an approximation to $V_\alpha(i)$. If g_{n+1} is the stationary policy that takes that action in state i minimizing
$$C(i, a) + \alpha \sum_{j=0}^{m} P_{ij}(a)(T_\alpha^n u)(j),$$

then g_{n+1} can be used as an approximation to the α-optimal policy. Furthermore, if n is large enough, then an α-optimal policy will be obtained (see Problem 13.6). Since the method of successive approximations does not specify how large n needs to be, one alternative is to compute $V_{g_{n+1}, \alpha}$ for various values of n. If $V_{g_{n+1}, \alpha}$ satisfies Eq. (13.8), then policy g_{n+1} is α-optimal.

If we let $u(i) = 0$ for all $i \in I_m$, then it is easy to show that $(T_\alpha^n 0)(i)$ is equal to $V_\alpha(i, n)$, the minimum expected discounted cost for an n-period problem given that you start in state i. Thus,

$$\lim_{n \to \infty} V_\alpha(i, n) = V_\alpha(i).$$

Problem 13.2. Show that $|V_\alpha(i, n) - V_\alpha(i)| < \alpha^n M/(1 - \alpha)$ for all $i \in I_m$.

As an example of the method of successive approximations, consider the above inventory problem. In Table 13.2 we give $V_\alpha(i, n)$ and $g_n(i)$ for various values of n (computed from recurrence relation (13.4)). Notice that the α-optimal policy is obtained for $n \geqslant 2$ and that $V_\alpha(i, n)$ differs from $V_\alpha(i)$ by less than 1% for $n = 50$.

One iteration of the method of successive approximations, computing the table $(T_\alpha u)(i)$, $i = 0, \ldots, m$, from the table $u(i)$, $i = 0, \ldots, m$, by (13.10), involves much less computation than one iteration of the policy improvement algorithm. Computing each number $(T_\alpha u)(i)$ by the method of successive approximations is nothing more than the usual dynamic-programming procedure of fixing the state, i, and then minimizing over all admissible decisions, using $u(j), j = 0, \ldots, m$, on the r.h.s. to play the role of the remaining expected cost after the first transition. This is done successively for $i = 0, 1, \ldots, m$. The policy improvement algorithm on the other hand, requires at each iteration the much more time-consuming solution of a system of $m + 1$ equations in $m + 1$ unknowns.

In addition to the policy improvement algorithm and the method of successive approximations, the problems considered in this section can also

Table 13.2. *Solution by the method of successive approximations*

n	$V_\alpha(0, n)$	$V_\alpha(1, n)$	$V_\alpha(2, n)$	$V_\alpha(3, n)$	$g_n(0)$	$g_n(1)$	$g_n(2)$	$g_n(3)$
1	10.00	8.00	2.00	1.38	2	1	0	0
2	16.71	14.71	9.65	6.71	3	2	0	0
3	22.80	20.80	15.79	12.80	3	2	0	0
10	51.45	49.45	44.44	41.45	3	2	0	0
30	74.53	72.53	67.52	64.53	3	2	0	0
50	77.34	75.34	70.33	67.34	3	2	0	0
∞	77.73	75.73	70.71	67.73	3	2	0	0

be solved by using linear programming. However, we shall not consider that technique here. See, for example, Derman [1].

Problem 13.3. At the beginning of each month a machine is inspected and classified into one of the following states:

State	Interpretation
0	good as new
1	operable—minor deterioration
2	operable—major deterioration
3	inoperable

After observing the state of the machine, a decision must be made whether to keep the machine for another month (action K) or to replace the current machine by a new machine (action R). However, in state 0 only action K is allowable and in state 3 only action R is allowable. If action K is taken, then monthly operating costs of 100, 200, and 500 are incurred when in states 0, 1, and 2, respectively. Furthermore, the machine will be in state j at the beginning of the next month with probability $P_{ij}(K)$, $i = 0, 1, 2$. If action R is taken, then the machine is instantaneously replaced by a new machine (state 0) and replacement costs of 2000, 2500, and 3000 are incurred when in states 1, 2, and 3, respectively. The machine will be in state j at the beginning of the next month with probability $P_{ij}(R) = P_{0j}(K)$, $i = 1, 2, 3$. The $C(i, a)$ and $P_{ij}(a)$ for this problem are given in Table 13.3. Use the policy improvement algorithm with $\alpha = 0.95$ to find an optimal replacement policy.

Problem 13.4. Steffi would like to sell her house. At the beginning of each week she receives an offer, which must be accepted (action A) or rejected (action R) immediately. Once an offer is rejected, it is lost forever. Assume that successive offers are independent of each other and take on the values 38,000, 40,000, and 42,000 with respective probabilities $\frac{1}{2}$, $\frac{3}{8}$, and $\frac{1}{8}$. Also assume that a maintenance cost of 50 is incurred each week that the house remains unsold, and that $\alpha = 0.99$. Let states 0, 1, and 2, respectively, correspond to the three possible offers. In addition assume that the process goes to state ∞ once an offer is accepted, and stays there indefinitely at a weekly cost of 0. Use the policy improvement algorithm to find the policy that will *maximize* Steffi's expected total discounted profit.

Table 13.3. *Data for Problem 13.3*

State i	Action a	$C(i, a)$	$P_{i0}(a)$	$P_{i1}(a)$	$P_{i2}(a)$	$P_{i3}(a)$
0	K	100	.7500	.1875	.0625	0
1	K	200	0	.7500	.1875	.0625
	R	2100	.7500	.1875	.0625	0
2	K	500	0	0	.7500	.2500
	R	2600	.7500	.1875	.0625	0
3	R	3100	.7500	.1875	.0625	0

Problem 13.5. Use three iterations of the method of successive approximations (with $u(i) = 0$ for $i \in I_m$) to approximate the optimal solution for Problem 13.3.

Problem 13.6. For the method of successive approximations (with $u(i) = 0$ for all $i \in I_m$), show that there exists an n^* such that g_n is α-optimal for $n > n^*$. (A similar proof is valid for any $u \in B(I_m)$.)

References

[1] Derman, C., *Finite State Markovian Decision Processes*, Academic Press, New York, 1970.

[2] Howard, R. A., *Dynamic Programming and Markov Processes*, M.I.T. Press, Cambridge, Massachusetts, 1960.

[3] Ross, S. M., *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, 1970.

Chapter 14

STOCHASTIC PROBLEMS WITH LINEAR DYNAMICS AND QUADRATIC CRITERIA

1. Introduction

We return to the model of Chapter 6. First we consider a rather special stochastic version of the model where there is an additive random term in the linear dynamics. The problem is solved by only slight modification of the methods developed in Chapter 6. It turns out that the optimal control policy is the same as one obtains for a related deterministic problem, a result called *certainty equivalence*. The model is particularly appropriate to stochastic inventory problems (see Chapter 11) where the inventory starting stage $i + 1$ is the inventory starting stage i (the state) plus the amount acquired (the decision) minus the demand (a random variable).

We next consider a very general linear dynamics, quadratic criterion model in which all the data are random and find that the methods of Chapter 6 still apply.

There is no discussion here, such as in Section 5 of Chapter 6, of fixed terminal states since, for stochastic dynamics, the terminal state cannot be specified with certainty.

2. Certainty Equivalence

We study the problem: choose the feedback policy $y(0)$, $y(1)$, $y(2), \ldots, y(N-1)$, where $y(1), \ldots, y(N-1)$ depend on $x(1), \ldots, x(N-1)$, respectively, that minimizes the expected value of J where J is given by

$$J = \sum_{i=0}^{N-1} \left[a(i)x^2(i) + c(i)y^2(i) \right] + lx^2(N), \qquad (14.1)$$

188

given the stochastic dynamical rule

$$x(i + 1) = g(i)x(i) + h(i)y(i) + z(i), \qquad x(0) \text{ specified,} \quad (14.2)$$

where $z(i)$ is a random variable, independently chosen at each stage i from a distribution with mean $\bar{z}(i)$ and variance $\sigma^2(i)$. We assume in the text that $y(i)$ is chosen *before* the random variable $z(i)$ is observed. (See Problem 14.1 for a different assumption.)

We achieve · the dynamic-programming solution by defining the optimal expected value function by

$$
\begin{aligned}
V_i(x) = \ &\text{the minimum expected cost of the remaining process} \\
&\text{if we start stage } i \text{ in state } x \text{ and have not yet learned} \\
&\text{the actual value of } z(i). \qquad\qquad\qquad\qquad\qquad (14.3)
\end{aligned}
$$

Then, by the principle of optimality, where E_z denotes the expected value taken with respect to the random variable z,

$$V_i(x) = \min_y \left\{ \mathop{E}_{z(i)} \left[a(i)x^2 + c(i)y^2 + V_{i+1}(g(i)x + h(i)y + z(i)) \right] \right\};$$
$$(14.4)$$

i.e., for each possible choice of y, given i and x, we compute the expected cost of the remaining process with respect to the random variable $z(i)$ where the next state is a random variable depending on $z(i)$, and then we choose that y that minimizes the expected remaining cost. The boundary condition is

$$V_N(x) = lx^2. \qquad (14.5)$$

Problem 14.1. Suppose that at each stage i we choose $y(i)$ *after* we are told the value of the random variable $z(i)$. The problem is still a stochastic one because we do not know $z(i + 1), \ldots, z(N - 1)$ when we are asked to choose $y(i)$. (See Problem 9.11.) Write the appropriate versions of (14.3)–(14.5).

We now use (14.4) and (14.5) to determine the function $V_{N-1}(x)$ analytically:

$$
\begin{aligned}
V_{N-1}(x) = \min_y \Big\{ \ & \mathop{E}_{z(N-1)} \Big[a(N-1)x^2 + c(N-1)y^2 \\
& + V_N(g(N-1)x + h(N-1)y + z(N-1)) \Big] \Big\} \\
= \min_y \Big\{ \ & \mathop{E}_{z(N-1)} \Big[a(N-1)x^2 + c(N-1)y^2 \\
& + l(g(N-1)x + h(N-1)y + z(N-1))^2 \Big] \Big\} \\
= \min_y \Big\{ \ & \mathop{E}_{z(N-1)} \Big[a(N-1)x^2 + c(N-1)y^2 \\
& + l\big(g^2(N-1)x^2 + h^2(N-1)y^2 + z^2(N-1)
\end{aligned}
$$

equation continues

$$+ 2g(N-1)h(N-1)xy + 2g(N-1)xz(N-1)$$

$$+ 2h(N-1)yz(N-1))]\}$$

$$= \min_y \{ a(N-1)x^2 + c(N-1)y^2 + lg^2(N-1)x^2$$

$$+ lh^2(N-1)y^2 + l(\bar{z}^2(N-1) + \sigma^2(N-1))$$

$$+ 2lg(N-1)h(N-1)xy + 2lg(N-1)x\bar{z}(N-1)$$

$$+ 2lh(N-1)y\bar{z}(N-1)\} \tag{14.6}$$

In the last step above we have explicitly taken the expected value and have used the fact that if z is a random variable with mean $\bar{z}$ and variance σ^2,

$$E(z^2) = \bar{z}^2 + \sigma^2. \tag{14.7}$$

To derive (14.7) we note that by the definition of variance,

$$\sigma^2 = E\left[(z - \bar{z})^2\right] \tag{14.8}$$

and hence, recognizing that $\bar{z}$ is not a random variable but rather a given number,

$$\sigma^2 = E(z^2 - 2z\bar{z} + \bar{z}^2) = E(z^2) - 2\bar{z}E(z) + \bar{z}^2$$

$$= E(z^2) - 2\bar{z}^2 + \bar{z}^2 = E(z^2) - \bar{z}^2, \tag{14.9}$$

which gives (14.7).

The y that minimizes the bracketed expression in (14.6) is found by calculus as follows:

$$0 = 2c(N-1)y + 2lh^2(N-1)y + 2lg(N-1)h(N-1)x$$

$$+ 2lh(N-1)\bar{z}(N-1),$$

$$y = - \frac{lg(N-1)h(N-1)x + lh(N-1)\bar{z}(N-1)}{c(N-1) + lh^2(N-1)}, \tag{14.10}$$

where we assume that the denominator (which is the second derivative with respect to y) is positive.

Substitution of this minimizing y into (14.6) and some algebra yields the following quadratic expression for $V_{N-1}(x)$:

$$V_{N-1}(x) = \left[a(N-1) + lg^2(N-1) - \frac{l^2h^2(N-1)g^2(N-1)}{c(N-1) + lh^2(N-1)} \right]x^2$$

$$+ \left[2lg(N-1)\bar{z}(N-1) - \frac{2l^2h^2(N-1)g(N-1)\bar{z}(N-1)}{c(N-1) + lh^2(N-1)} \right]x$$

$$+ \left[l\bar{z}^2(N-1) + l\sigma^2(N-1) - \frac{l^2h^2(N-1)\bar{z}^2(N-1)}{c(N-1) + lh^2(N-1)} \right]. \tag{14.11}$$

Hence a quadratic form lx^2 at stage N has become a general quadratic function of the form $p(N - 1)x^2 + q(N - 1)x + r(N - 1)$ at stage $N - 1$.

We could now compute $V_{N-2}(x)$ using (14.4) with $i = N - 2$, and result (14.11). If we did, we would find that $V_{N-2}(x)$ was a quadratic function of the same form as $V_{N-1}(x)$, so we proceed instead to an inductive proof. Note that $V_N(x)$ is of the form

$$V_N(x) = p(N)x^2 + q(N)x + r(N) \tag{14.12}$$

with

$$p(N) = l, \qquad q(N) = 0, \qquad r(N) = 0. \tag{14.13}$$

Assume that $V_{i+1}(x)$ is of the form

$$V_{i+1}(x) = p(i + 1)x^2 + q(i + 1)x + r(i + 1). \tag{14.14}$$

We now show that in this case $V_i(x)$ has the form

$$V_i(x) = p(i)x^2 + q(i)x + r(i), \tag{14.15}$$

and that therefore V is always quadratic. While showing this, we obtain recurrence formulas for $p(i)$, $q(i)$, and $r(i)$ and a formula for $y(i)$ as a linear function of $x(i)$.

Proceeding a bit sketchily since no new ideas are introduced, and omitting the stage arguments on the data until the final results, we have

$$V_i(x) = \min_y \left\{ E_z \left[ax^2 + cy^2 + p(gx + hy + z)^2 + q(gx + hy + z) + r \right] \right\}$$
$$= \min_y \left[ax^2 + cy^2 + pg^2x^2 + ph^2y^2 + p(\bar{z}^2 + \sigma^2) + 2pghxy \right.$$
$$\left. + 2phy\bar{z} + 2pgx\bar{z} + qgx + qhy + q\bar{z} + r \right], \tag{14.16}$$

$$y = -\frac{2pghx + 2ph\bar{z} + qh}{2(c + ph^2)}, \tag{14.17}$$

$$V_i(x) = \left[a + pg^2 - \frac{p^2h^2g^2}{c + ph^2} \right]x^2 + \left[2pg\bar{z} + qg - \frac{pgh(2ph\bar{z} + qh)}{c + ph^2} \right]x$$
$$+ \left[p\bar{z}^2 + p\sigma^2 + q\bar{z} + r - \frac{(2ph\bar{z} + qh)^2}{4(c + ph^2)} \right]. \tag{14.18}$$

Hence $p(i)$, $q(i)$, and $r(i)$ satisfy the recurrence relations

$$p(i) = a(i) + p(i + 1)g^2(i) - \frac{p^2(i + 1)h^2(i)g^2(i)}{c(i) + p(i + 1)h^2(i)}, \tag{14.19}$$

$$q(i) = 2p(i + 1)g(i)\bar{z}(i) + q(i + 1)g(i)$$
$$- \frac{p(i + 1)g(i)h(i)\left[2p(i + 1)h(i)\bar{z}(i) + q(i + 1)h(i)\right]}{c(i) + p(i + 1)h^2(i)}, \tag{14.20}$$

$$r(i) = p(i + 1)\bar{z}^2(i) + p(i + 1)\sigma^2(i) + q(i + 1)\bar{z}(i) + r(i + 1)$$

$$- \frac{(2p(i + 1)h(i)\bar{z}(i) + q(i + 1)h(i))^2}{4(c(i) + p(i + 1)h^2(i))}, \qquad (14.21)$$

with boundary conditions (14.13), and the optimal decision $y(i)$ is given by

$$y(i) = - \frac{2p(i + 1)g(i)h(i)x(i) + 2p(i + 1)h(i)\bar{z}(i) + q(i + 1)h(i)}{2(c(i) + p(i + 1)h^2(i))}.$$

$$(14.22)$$

The above constitutes the solution of the problem with random additive term in the dynamics. An interesting connection between this solution and the solution to a certain deterministic problem will now be made. If the reader refers to the solution to the deterministic Problem 6.4 and sets $b(i) = d(i) = e(i) = f(i) = 0$ (as in the stochastic problem we have just solved) and, furthermore, if he replaces $k(i)$ by $\bar{z}(i)$, he will find that the formulas for $p(i)$ and $q(i)$, and for $y(i)$ in terms of p and q, are exactly the same as for the above stochastic problem. The formulas for $r(i)$ differ by a term $p(i + 1)\sigma^2(i)$, but $r(i)$ does not enter into the computation of the optimal decision $y(i)$. If $b, d, e,$ and f were not zero in our criterion (14.1), our results would still agree in the above sense with the results of Problem 6.4. What we can conclude, then, is that the optimal policy, as a function of state, is the same for the stochastic problem as for the deterministic problem constructed from the stochastic one by replacing the random variable by its expected value. The cost of the optimal policy differs due to the different formulas for $r(i)$, which reflects the cost of the randomness, the inability to be sure of getting what we want. The identity of optimal policies for the stochastic problem and the related deterministic problem constructed by replacing the random variable by its expected value is called *certainty equivalence*. This sort of result holds only in very special situations involving linearity. When it holds, it justifies replacing a stochastic problem by a presumably easier deterministic problem, but one cannot expect this sort of replacement to be valid in general. In fact, in the next section we make only a minor change in the above model, and certainty equivalence fails to hold.

Problem 14.2. Develop recursive formulas for Problem 14.1. Does certainty equivalence hold?

Problem 14.3. The effect on the expected cost of the randomness of z is an extra term $p(i + 1)\sigma^2(i)$ in the formula for r. Can you prove that the existence of uncertainty about z always leads to a larger expected cost than in the deterministic problem with $\bar{z}$ replacing z, or can you construct a problem where the uncertainty *reduces* the expected cost?

3. A More General Stochastic Model

We consider now the problem obtained from (14.1) and (14.2) by assuming that all data, $a(i)$, $c(i)$, l, $g(i)$, and $h(i)$ as well as $z(i)$ are random. We assume (though it is not necessary) that all these random variables are independent of each other at a given stage, and also that they are independent from stage to stage. Let $\bar{a}(i)$, $\bar{c}(i)$, etc., denote means and $\sigma_a^2(i)$, $\sigma_c^2(i)$, etc., denote variances.

We define the optimal expected value function by

$V_i(x)$ = the minimum expected cost of the remaining process
if we start stage i in state x and have not yet learned
the actual values of the random variables at that
stage.

Then, letting $B(i)$ denote the set of all random variables at stage i, we have

$$V_i(x) = \min_y \left\{ \underset{B(i)}{E} \left[a(i)x^2 + c(i)y^2 + V_{i+1}(g(i)x + h(i)y + z(i)) \right] \right\}$$

$$(14.23)$$

with boundary condition

$$V_N(x) = \bar{l}x^2. \qquad (14.24)$$

Based on the previous section, we immediately make the inductive hypothesis that

$$V_{i+1}(x) = u(i+1)x^2 + v(i+1)x + w(i+1), \qquad (14.25)$$

which holds when $i + 1$ equals N, with

$$u(N) = \bar{l}, \qquad v(N) = 0, \qquad w(N) = 0. \qquad (14.26)$$

Problem 14.4.   Now substitute (14.25) into (14.23) to prove the inductive hypothesis and obtain recursive formulas for $u(i)$, $v(i)$, and $w(i)$ as well as a formula for $y(i)$ in terms of $x(i)$, u, v, and w. Compare your results with (14.19)–(14.22). Does certainty equivalence hold?

The important result here is that the quite general stochastic problem with linear dynamics and quadratic criterion can easily be solved, even for 10 or 20 state variables, by the formulas just derived by the reader. Essentially no more computation is required than for the deterministic problem.

Problem 14.5.   Consider the linear dynamics, quadratic criterion problem:

$$\min \sum_{i=0}^{N-1} (x^2(i) + y^2(i)) + x^2(N),$$

$$x(i + 1) = x(i) + y(i), \qquad x(0) = x_0,$$

where all you know up until stage N_3 is that the duration N is N_1 with probability v and is N_2 $(< N_1)$ with probability $1 - v$. You find out at stage N_3 $(< N_2)$, before making your decision at stage N_3, what the true duration is to be. Give an efficient solution procedure paying special attention at stage $N_3 - 1$ and stage N_3.

Problem 14.6. Consider the stochastic dynamical system

$$x(i + 1) = gx(i) + hy(i) + z(i), \qquad x(0) = x_0,$$

where $z(i)$ is a random variable with mean 0 and variance $\sigma^2 y^2(i)$ (i.e., the larger $|y(i)|$, the more "randomness") with $z(i)$ independent of $z(j)$ for $i \neq j$. A feedback control policy (where y is chosen at stage i based on knowledge of $x(i)$ before $z(i)$ is observed) is to be determined that minimizes the expected value of

$$\sum_{i=0}^{N-1} (ax^2(i) + cy^2(i)) + lx^2(N).$$

Develop formulas for solving the problem.

Problem 14.7. Consider the following stochastic linear dynamical system with time lag in the effect of decision y,

$$x(i + 1) = gx(i) + hy(i - 1) + z(i),$$

where $z(i)$ is a random variable with $E(z(i)) = 0$, $E(z^2(i)) = \sigma^2$, $z(i)$ and $z(j)$ independent for $i \neq j$. The decision $y(i)$ is chosen with $x(i)$ and $y(i - 1)$ known but $z(j)$, $j = i, i + 1, \ldots, N - 1$, not yet known, but it has no effect until one period later. The process starts with given $x(1)$ and $y(0)$. The $y(i)$, $i = 1, \ldots, N - 2$, are to be chosen so as to minimize the expected value of the quadratic criterion J given by

$$J = \sum_{i=1}^{N-2} (x^2(i) + y^2(i)) + x^2(N - 1) + x^2(N).$$

Give a dynamic-programming solution procedure.

Chapter 15

OPTIMIZATION PROBLEMS INVOLVING LEARNING

1. Introduction

In Chapter 9, when we introduced risk into our previously deterministic models, we said that really accurate representations of reality generally involve learning. Sometimes the learning is active. Decisions are made as much to acquire information and understanding as to achieve gratification. Sampling surveys in business, or the dating process in socialization, are examples of probing for information. Other times the learning is passive. In designing an inventory policy for a new item, time brings a better appreciation of the demand pattern for the item and this affects the policy, but the acquisition of this information does not depend on our decisions; so our learning, while useful, is passive.

The decision theory of this chapter is frequently referred to as *adaptive control* theory. Sometimes it is called *dual · control* theory to emphasize the concurrent concern with learning and with optimization. A special model involving a structuring based on alternating choice and chance outcome is called *statistical decision theory* or *decision analysis*. The learning model is also occasionally called decision making under uncertainty, to distinguish it from stochastic problems, called decision making under risk. Another name is *Bayesian decision theory* since the approach makes heavy use of a result from probability theory called Bayes' law.

As the reader will soon see, the incorporation of learning into a dynamic-programming model entails an increase in the number of state variables. This usually precludes computational or analytical solution. A

realistic model, incapable of solution, is no more praiseworthy than an unrealistic, solved, problem. For this reason, stochastic dynamic-programming models are probably the most important.

We treat first a simple shortest-path problem involving learning. While the basic idea is simple, even after making numerous simplifying assumptions we are led to a formulation involving enough computation to require a digital computer.

Before presenting the shortest-path problem and other applications, however, we shall explain a result from probability theory called Bayes' law which underlies our approach to learning.

2. Bayes' Law

Let each one of n events A_i, $i = 1, \ldots, n$, have probability $P(A_i)$ of occurring. Exactly one of the events has occurred, but which one is unknown to us. Suppose that we have observed that a related event B has occurred and that $P(B|A_i)$, the probability of event B occurring given that A_i has occurred, is known for all i. Then, the probability that event A_i has occurred, given the observed event B, is written $P(A_i|B)$ and is given by the formula, called Bayes' law,

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{n} P(B|A_j)P(A_j)} . \qquad (15.1)$$

The denominator on the right is $P(B)$, the probability of event B occurring, and if we multiply both sides of (15.1) by $P(B)$ the two sides of the resulting equation are just two different ways of computing $P(A_i, B)$, the probability of the joint event A_i and B both occurring.

Problem 15.1. Suppose that we have two coins. Coin 1 has $P(\text{heads}) = \frac{3}{4}$ and $P(\text{tails}) = \frac{1}{4}$ and coin 2 has $P(\text{heads}) = \frac{1}{4}$ and $P(\text{tails}) = \frac{3}{4}$. A coin is selected at random and flipped, and comes up heads. What is the probability that the coin that was flipped is coin 1?

Problem 15.2. What is the probability that the coin that was flipped and came up heads in Problem 15.1 is coin 1 if the coin to be flipped was determined by rolling a die and if it showed 1 or 2 coin 1 was flipped, while if it showed 3, 4, 5, or 6, coin 2 was flipped. We know the rule by which the flipped coin was chosen, but not what the die actually showed.

A continuous version of Bayes' law is as follows: Let $f(x|y)$, $a \leqslant x \leqslant b$, be the density function of a continuous random variable given

that a related discrete variable has value y; i.e., $f(x|y)\,dx$ is the probability that the value of the continuous variable is between x and $x + dx$, given y. Let $g(y|x)\,dx$ be the probability of the discrete variable equaling y given that the continuous variable is between x and $x + dx$. Let $h(x)\,dx$ be the probability, before observing y, that the continuous variable is between x and $x + dx$. Then

$$f(x|y) = \frac{g(y|x)h(x)}{\int_a^b g(y|z)h(z)\,dz}. \tag{15.2}$$

Problem 15.3. A coin is weighted so that its probability of heads is p, where p is determined by sampling a uniform distribution; i.e., the density function of p equals 1 and is defined between 0 and 1. We know how p was determined, but not its value. We flip the coin k times and observe l heads. What is the density function of the random variable p based on our observation? What is $\bar{p}$, the expected value of p, after k flips have yielded l heads?

Bayes' law tells us how to adjust the probability density of an event that we cannot observe directly, once the event has occurred and an event that is related to the occurred event is observed. The probability density describing the situation before the observation is called the *prior* density and the density as computed by Bayes' law after observing the related event is called the *posterior* density. The use of an observed event to modify our uncertainty about a situation constitutes learning, and Bayes' law is the backbone of an optimization theory involving learning.

3. A Shortest-Path Problem with Learning

In Chapter 1 we discussed acyclic shortest-path problems in which we could choose, with certainty, at each decision point the arc that we wished to follow. Such processes are called deterministic. In Chapter 9 we reconsidered similar problems, but assumed that our decision at a vertex determined only the probabilities of our following various arcs. We knew the probabilities associated with each decision. We called this a stochastic path problem. Now we return once more to the path problem. We now assume that each decision at each vertex determines the probabilities of following various arcs, but that we do not know the probabilities associated with the decisions. However, by making decisions and observing transitions we learn about the fixed, but unknown, probabilities.

The problem is represented by Figure 15.1. We wish to minimize the expected cost of moving from A $((0, 0))$ to line B $(i = 5)$. At each vertex we
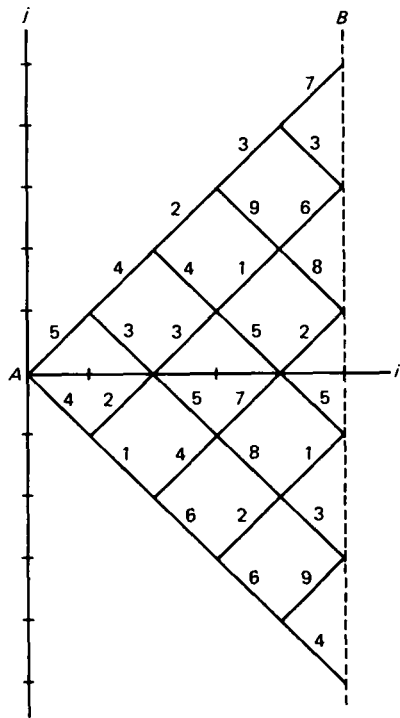
Figure 15.1

must choose one out of two possible decisions. Decision 1 at vertex (i, j) results in a move to vertex $(i + 1, j + 1)$ with probability p_1 and in a move to $(i + 1, j - 1)$ with probability $1 - p_1$. p_1 is a fixed number, held constant throughout the entire problem, but we do not know its value. We do know that p_1 was determined by generating a random number between 0 and 1 with $f_1(x) \, dx$ equaling the probability that $x \leqslant p_1 \leqslant x + dx$. Hence $f_1(x)$ is what we called the prior density of p_1. After choosing decision 1 one or more times and observing the transitions that occur, we can use Bayes' law to calculate the posterior density of p_1. A description of decision 2 parallels that of decision 1 except that p_2 replaces p_1, and the prior density of p_2 is $f_2(x)$.

There are two competing motives for choosing a particular decision at a given vertex. The decision will result in a transition with an associated cost and observation of the transition resulting from the decision will also yield further information about the value of p_i ($i = 1$ or 2) for that decision. The learning process is active in that we choose which decision we wish to learn more about.

If we were hired as a consultant to someone in the act of solving such

a path problem, we would want to know the current vertex and also the current state of knowledge about p_1 and p_2. The probability density for p_1 depends on the prior probability density $f_1(x)$ and on the number of times decision 1 has been chosen and on the number of resulting upward and downward transitions. By Bayes' law, if decision 1 has been chosen k_1 times with l_1 upward transitions (and $k_1 - l_1$ downward transitions) resulting, we can compute $f_1(x|k_1, l_1)$, the posterior density for p_1, obtaining

$$f_1(x|k_1, l_1) = \frac{x^{l_1}(1 - x)^{k_1 - l_1}f_1(x)}{\int_0^1 z^{l_1}(1 - z)^{k_1 - l_1}f_1(z)\, dz}. \tag{15.3}$$

Likewise, if decision 2 has been chosen k_2 times with l_2 upward (and $k_2 - l_2$ downward) transitions resulting, the posterior density for p_2, $f_2(x|k_2, l_2)$, is given by

$$f_2(x|k_2, l_2) = \frac{x^{l_2}(1 - x)^{k_2 - l_2}f_2(x)}{\int_0^1 z^{l_2}(1 - z)^{k_2 - l_2}f_2(z)\, dz}. \tag{15.4}$$

Knowing that the process started at $(0, 0)$, if we are told k_1, l_1, k_2, and l_2, we can compute the current vertex (i, j) by

$$i = k_1 + k_2, \qquad j = l_1 + l_2 - (k_1 - l_1) - (k_2 - l_2) = 2(l_1 + l_2) - (k_1 + k_2). \tag{15.5}$$

Consequently, the state variables (from which we can deduce the stage i) are k_1, l_1, k_2, and l_2. We define

$S(k_1, l_1, k_2, l_2) =$ the minimum expected cost of the remaining process if we start at the vertex (i, j) given by (15.5) and k_1, l_1, k_2, and l_2 are as defined above. $\tag{15.6}$

If we choose decision 1 and p_1 is equal to x, we move upward with probability x, incurring cost $a_u(i, j)$ ($a_u(i, j)$ is, as before, the cost of the diagonally upward arc from (i, j) and (i, j) is given by (15.5)) and moving to the state $(k_1 + 1, l_1 + 1, k_2, l_2)$. With probability $1 - x$ we move diagonally downward, at cost $a_d(i, j)$, arriving at the state $(k_1 + 1, l_1, k_2, l_2)$. Hence, if $p_1 = x$ and we choose decision 1 and then continue optimally from the new state, the expected cost $E_1(x)$ is

$$E_1(x) = x\left[a_u(i, j) + S(k_1 + 1, l_1 + 1, k_2, l_2)\right]$$
$$+ (1 - x)\left[a_d(i, j) + S(k_1 + 1, l_1, k_2, l_2)\right].$$

But p_1 is between x and $x + dx$ with probability $f_1(x|k_1, l_1)\, dx$ where f_1 is

given by (15.3). Hence our expected cost, knowing only the density for p_1, is

$$\int_0^1 E_1(x) f_1(x|k_1, l_1) \, dx$$

$$= \int_0^1 \{ x[a_u(i, j) + S(k_1 + 1, l_1 + 1, k_2, l_2)]$$

$$+ (1 - x)[a_d(i, j) + S(k_1 + 1, l_1, k_2, l_2)]\} f_1(x|k_1, l_1) \, dx.$$

Reasoning likewise for decision 2, we have the recurrence relation

$$S(k_1, l_1, k_2, l_2)$$

$$= \min \begin{bmatrix} 1: & \int_0^1 \{ x[a_u(i, j) + S(k_1 + 1, l_1 + 1, k_2, l_2)] \\ & + (1 - x)[a_d(i, j) \\ & + S(k_1 + 1, l_1, k_2, l_2)]\} f_1(x|k_1, l_1) \, dx \\ 2: & \int_0^1 \{ x[a_u(i, j) + S(k_1, l_1, k_2 + 1, l_2 + 1)] \\ & + (1 - x)[a_d(i, j) \\ & + S(k_1, l_1, k_2 + 1, l_2)]\} f_2(x|k_2, l_2) \, dx \end{bmatrix}, \qquad (15.7)$$

where (i, j) is given by (15.5) and f_1 and f_2 by (15.3) and (15.4). The boundary condition is

$$S(k_1, l_1, k_2, l_2) = 0 \qquad \text{for all} \quad l_1 \leqslant k_1 \text{ and } l_2 \leqslant k_2 \quad \text{when } k_1 + k_2 = 5. \tag{15.8}$$

We solve recursively, first for all pairs of nonnegative integers k_1 and k_2 that sum to 4 and all $l_1 \leqslant k_1$, $l_2 \leqslant k_2$, then for all k_1 and k_2 summing to 3, then 2, then 1, and finally we compute the answer $S(0, 0, 0, 0)$.

Equation (15.7) can be simplified by writing

$$\bar{p}_1(k_1, l_1) = \int_0^1 x f_1(x|k_1, l_1) \, dx, \qquad \bar{p}_2(k_2, l_2) = \int_0^1 x f_2(x|k_2, l_2) \, dx,$$

where $\bar{p}_1$ and $\bar{p}_2$ are the means of the densities f_1 and f_2. Then (15.7) becomes

$$S(k_1, l_1, k_2, l_2)$$

$$= \min \begin{bmatrix} 1: & \bar{p}_1(k_1, l_1)[a_u(i, j) + S(k_1 + 1, l_1 + 1, k_2, l_2)] \\ & + (1 - \bar{p}_1(k_1, l_1))[a_d(i, j) + S(k_1 + 1, l_1, k_2, l_2)] \\ 2: & \bar{p}_2(k_2, l_2)[a_u(i, j) + S(k_1, l_1, k_2 + 1, l_2 + 1)] \\ & + (1 - \bar{p}_2(k_2, l_2))[a_d(i, j) + S(k_1, l_1, k_2 + 1, l_2)] \end{bmatrix}. \tag{15.9}$$

If we take the prior densities $f_1(x)$ and $f_2(x)$ to be uniform, then, by the solution of Problem 15.3, we can further simplify (15.9), obtaining,

$S(k_1, l_1, k_2, l_2)$

$$
= \min \begin{bmatrix}
1: & \dfrac{l_1 + 1}{k_1 + 2} \left[a_u(i, j) + S(k_1 + 1, l_1 + 1, k_2, l_2) \right] \\[2ex]
& + \left(1 - \dfrac{l_1 + 1}{k_1 + 2} \right) \left[a_d(i, j) + S(k_1 + 1, l_1, k_2, l_2) \right] \\[2ex]
2: & \dfrac{l_2 + 1}{k_2 + 2} \left[a_u(i, j) + S(k_1, l_1, k_2 + 1, l_2 + 1) \right] \\[2ex]
& + \left(1 - \dfrac{l_2 + 1}{k_2 + 2} \right) \left[a_d(i, j) + S(k_1, l_1, k_2 + 1, l_2) \right]
\end{bmatrix},
$$

(15.10)

where, as before, i and j are given by (15.5).

Table 15.1 gives the computational solution of recurrence relation (15.10). Note how, when learning is present, the number of state variables increases and even the simplest path problem becomes more suitable for the digital computer than for hand calculation.

The optimal policy given in Table 15.1 calls for either decision 1 or 2 at (0, 0) (which is expected since both p_1 and p_2 have been generated by sampling from the same, uniform density and we have learned nothing further). If decision 1 is chosen and leads to a diagonally upward transition, the state is (1, 1, 0, 0), we are at the vertex (1, 1), and the optimal decision is 2. If the initial transition after decision 1 at (0, 0) is downward, the state is (1, 0, 0, 0) and decision 1 should be chosen. If in this latter case, the transition is upward, the state is (2, 1, 0, 0), we are at the vertex (2, 0), and decision 2 is optimal. This is an interesting case since both $\bar{p}_1$ and $\bar{p}_2$ equal $\frac{1}{2}$, but more information can be gained from decision 2, which has never been chosen, than decision 1, which has already been chosen twice.

To review, processes involving learning require additional state variables. These variables summarize the information acquired in such a way as to allow the use of Bayes' law to compute posterior probability densities describing the current state of knowledge. Once these additional variables are identified and incorporated in the definition of the optimal expected value function, the usual reasoning of dynamic programming leads to a recurrence relation characterizing the optimal value and optimal policy functions.

Table 15.1   *The minimum expected cost function S and optimal policy function u computed using* (15.10)

	k_1	l_1	k_2	l_2	j	$S(k_1, l_1, k_2, l_2)$	$u(k_1, l_1, k_2, l_2)$
$i = k_1 + k_2 = 4$	4	4	0	0	4	5.0000	2
	4	3	0	0	2	6.6667	1
	4	2	0	0	0	3.5000	1 or 2
	4	1	0	0	− 2	2.0000	2
	4	0	0	0	− 4	4.8333	1
	3	3	1	1	4	5.6667	2
	3	2	1	1	2	6.6667	2
	3	1	1	1	0	3.0000	2
	3	0	1	1	− 2	1.6667	2
	3	3	1	0	2	6.4000	1
	3	2	1	0	0	3.2000	1
	3	1	1	0	− 2	2.2000	1
	3	0	1	0	− 4	5.0000	1
	2	2	2	2	4	6.0000	1 or 2
	2	1	2	2	2	6.5000	2
	2	0	2	2	0	2.7500	2
	2	2	2	1	2	6.5000	1
	2	1	2	1	0	3.5000	1 or 2
	2	0	2	1	− 2	2.0000	2
	2	2	2	0	0	2.7500	1
	2	1	2	0	− 2	2.0000	1
	2	0	2	0	− 4	5.2500	1 or 2
	1	1	3	3	4	5.6667	1
	1	0	3	3	2	6.4000	2
	1	1	3	2	2	6.6667	1
	1	0	3	2	0	3.2000	2
	1	1	3	1	0	3.0000	1
	1	0	3	1	− 2	2.2000	2
	1	1	3	0	− 2	1.6667	1
	1	0	3	0	− 4	5.0000	2
	0	0	4	4	4	5.0000	1
	0	0	4	3	2	6.6667	2
	0	0	4	2	0	3.5000	1 or 2
	0	0	4	1	− 2	2.0000	1
	0	0	4	0	− 4	4.8333	2
$i = k_1 + k_2 = 3$	3	3	0	0	3	9.5333	1
	3	2	0	0	1	7.9333	2
	3	1	0	0	− 1	10.1000	2
	3	0	0	0	− 3	7.3333	2
	2	2	1	1	3	10.4167	1
	2	1	1	1	1	7.8333	1 or 2
	2	0	1	1	− 1	9.7500	1
	2	2	1	0	1	7.6000	1
	2	1	1	0	− 1	10.1667	2
	2	0	1	0	− 3	8.8333	2
	1	1	2	2	3	10.4167	2

Table 15.1 (*cont.*)

	k_1	l_1	k_2	l_2	j	$S(k_1, l_1, k_2, l_2)$	$u(k_1, l_1, k_2, l_2)$
$i = k_1 + k_2 = 3$	1	0	2	2	1	7.6000	2
	1	1	2	1	1	7.8333	1 or 2
	1	0	2	1	-1	10.1667	1
	1	1	2	0	-1	9.7500	2
	1	0	2	0	-3	8.8333	1
	0	0	3	3	3	9.5333	2
	0	0	3	2	1	7.9333	1
	0	0	3	1	-1	10.1000	1
	0	0	3	0	-3	7.3333	1
$i = k_1 + k_2 = 2$	2	2	0	0	2	11.6333	1
	2	1	0	0	0	13.0000	2
	2	0	0	0	-2	13.5250	1
	1	1	1	1	2	12.2222	1 or 2
	1	0	1	1	0	12.1222	2
	1	1	1	0	0	12.1222	1
	1	0	1	0	-2	14.6111	1 or 2
	0	0	2	2	2	11.6333	2
	0	0	2	1	0	13.0000	1
	0	0	2	0	-2	13.5250	2
$i = k_1 + k_2 = 1$	1	1	0	0	1	15.6722	2
	1	0	0	0	-1	14.6833	1
	0	0	1	1	1	15.6722	1
	0	0	1	0	-1	14.6833	2
$i = k_1 + k_2 = 0$	0	0	0	0	0	19.6778	1 or 2

Problem 15.4. Suppose, in the problem of this section, that both p_1 and p_2 are determined by independently sampling a discrete distribution where $\text{Prob}(p_i = \frac{1}{4}) = \frac{1}{2}$, $\text{Prob}(p_i = \frac{3}{4}) = \frac{1}{2}$ for $i = 1$ and $i = 2$. Give the appropriate dynamic-programming formulation. Do not solve numerically.

Problem 15.5. Suppose, in Problem 15.4, that you know the actual values of p_1 and p_2 generated by the sampling but that the decision maker does not. How much should you charge him for this information so that his expected value of the information equals his cost? Give the answer in terms of certain values of certain optimal value functions.

4. A Quality Control Problem

We turn now to a problem that uses many of the concepts and formulas of the path problem, but is somewhat more practical. Suppose that a batch of n items has just been produced and paid for and that each item has a probability p of being good and a probability $1 - p$ of being

defective. While we do not know the value of p, based on experience with many previous batches we start with a prior probability density function for p, $f(p)$, $0 \leqslant p \leqslant 1$. We can learn about the true value of p for the batch on hand by examining the items. We propose to examine the items, one by one. Examination of an item costs c. Items found good upon examination are accepted and each has a value d_1. Items found to be defective are discarded at no cost beyond the cost of examination. At any point in the process, based on what we have learned about p, we may elect to accept all the remaining items rather than examine further. If we do so, each good item is worth d_1 as above, but each defective item accepted entails a cost d_2, usually large. We may also, at any point in the process, elect to reject all the remaining items. These rejected items have no value to us, nor any cost beyond the production cost, assumed already paid. Our problem is to decide when to stop examining individual items and what to do with the remaining items so as to maximize the expected value of the batch.

This problem occurs in production quality control and in the testing of medical drugs before marketing.

We begin by defining the optimal expected value function by

$S(k, l) =$ the maximum expected value of the remaining items in the batch if we have examined k items so far, and found l to be good.

Clearly, we have the boundary condition

$$S(n, l) = 0 \qquad \text{for} \quad 0 \leqslant l \leqslant n \tag{15.11}$$

since no items remain. We can compute, as in Eq. (15.3), the probability density of p, the probability that each individual item is good given k and l, by Bayes' law obtaining

$$f(p|k, l) = \frac{p^l(1-p)^{k-l}f(p)}{\int_0^1 z^l(1-z)^{k-l}f(z)\, dz}.$$

If we knew that p equaled x, the expected remaining value $E_1(x)$ if we examine one item and proceed optimally thereafter is given by

$$E_1(x) = -c + x[d_1 + S(k+1, l+1)] + (1-x)S(k+1, l).$$

Since we only know $f(p|k, l)$, but not the actual value of p, we multiply $E_1(x)$ by $f(x|k, l)$ and integrate over x to obtain E_2, the expected remaining value given our uncertainty about p. Hence

$$E_2 = -c + \int_0^1 \{x[d_1 + S(k+1, l+1)]$$
$$+ (1-x)[S(k+1, l)]\}f(x|k, l)\, dx$$
$$= -c + \bar{p}(k, l)[d_1 + S(k+1, l+1)] + (1 - \bar{p}(k, l))S(k+1, l),$$

where $\bar{p}(k, l)$ is the expected value of p given $f(p)$ and the observed k and l.

If we knew that p equaled x and decide to accept the remaining items after examining k items, $E_3(x)$, the expected value, is given by

$$E_3(x) = \sum_{i=0}^{n-k} \binom{n-k}{i} x^i (1-x)^{n-k-i} [d_1 i - d_2(n-k-i)]$$

$$= d_1(n-k)x - d_2[n-k-(n-k)x], \qquad (15.12)$$

where to obtain line one of (15.12) we have multiplied the value if i items of the remaining $n-k$ are good and $n-k-i$ are defective by the probability that i are good, and then summed over i. To obtain line two of (15.12) we have used the fact that the mean of the binomial distribution is $(n-k)x$. Since we do not know p, the expected value of accepting the remaining items, E_4, is given by

$$E_4 = (n-k)\int_0^1 \{d_1 x - d_2(1-x)\} f(x|k, l) \, dx$$

$$= (n-k)[d_1 \bar{p}(k, l) - d_2(1 - \bar{p}(k, l))].$$

If we reject the remaining lot, its value is zero.

Hence, we have the recurrence relation

$$S(k, l) = \max \begin{bmatrix} \text{Examine:} & E_2 \\ \text{Accept:} & E_4 \\ \text{Reject:} & 0 \end{bmatrix}.$$

If we further assume that $f(p)$ is the uniform density $f(p) = 1$, using calculations from Problem 15.3 we have

$$\bar{p}(k, l) = (l+1)/(k+2)$$

so

$$S(k, l) = \max \begin{bmatrix} E: & -c + \dfrac{l+1}{k+2}[d_1 + S(k+1, l+1)] \\ & + \left(1 - \dfrac{l+1}{k+2}\right)S(k+1, l) \\ A: & (n-k)\left[d_1\dfrac{l+1}{k+2} - d_2\left(1 - \dfrac{l+1}{k+2}\right)\right] \\ R: & 0 \end{bmatrix}. \quad (15.13)$$

Recurrence relation (15.13) with boundary condition (15.11) solves the problem. The answer is $S(0, 0)$.

This sampling problem only makes sense if p is unknown, so the only state variables constitute information about p. Hence the problem is numerically solvable for batches of reasonable size.

It can be shown that if "accept the remaining items" is the optimal

decision in state (k, l), accept is also the optimal decision for all (k', l') for $k' \leq k$ and $l' \geq l$. Similarly, if "reject the remaining items" is the optimal decision in state (k, l), reject is also the optimal decision for all (k', l') for $k' \geq k$ and $l' \leq l$.

Problem 15.6. Intuitively justify the result in the paragraph above about the structure of the optimal solution.

Problem 15.7. Discuss the range of the variables k and l used in the numerical solution of (15.13). Explain how the results justified in Problem 15.6 can be used to reduce the computational requirements.

5.   Decision Analysis

We now treat a problem involving learning that entails a choice of an action, followed by a result that yields information to which the decision maker can then choose his reaction, etc. This model differs from the usual one in two respects, yet the ideas we have already developed still apply. First, the model assumes that we are seeking to aid an individual make what is *for him* the best decision, based on his personal assessment of the situation and his personal evaluation of the desirability of the various possible results. In the previous models the costs and values were generally economic and were considered objectively determined by the situation. Second, the possible decisions and possible outcomes are quite different at each point in the process and depend strongly on previous decisions and their outcomes. In most previous models we assumed the problem was repetitive, with only the duration of the remaining process and the values of what we called the state variables differing at each step.

This particular model is generally treated under the title "Statistical Decision Theory" or "Decision Analysis" and is explicated in detail in the paperback book "Decision Analysis. Introductory Lectures on Choices Under Uncertainty" by Howard Raiffa, which is easy and fascinating reading. We shall say only enough here to, we hope, whet the reader's appetite for the above excellent reference. We outline the technique by means of an example, which follows.

The world-famous hard rock band "The Dynamic Programmers" are to give one of their dynamic programs at the local concert hall and tickets are to go on sale at the box office at 9 A.M. on Monday. Bert is trying to decide if he should compete with the mob for a ticket, and, if so, at what time he should join the ticket line. While the D.P.'s have never performed live locally, various lesser groups such as "The Linear Programmers" (a straight band) have, and so Bert has some ideas about the public's likely ticket-seeking behavior. Furthermore, by going to the box office at various times and observing the line, Bert can learn more about the D.P.'s

popularity. For simplicity, we assume that he structures the problem as follows (the letters in parentheses are abbreviations that we shall subsequently use to denote the decisions and outcomes):

A. If, upon arrival at the box office, he finds a long line (L) and joins it (J), Bert believes that his chance of obtaining a ticket (T) is 3 in 10, with 7 chances in 10 of the concert selling out (SO) before he reaches the box office. If he finds a line of medium length (M) and joins it, he believes that he has three chances in four of getting a ticket. If he joins a short line (S) he is sure of getting a ticket.

B. Bert can go to the box office at either 9 A.M. Sunday ($9S$), 6 P.M. Sunday ($6S$), 3 A.M. Monday ($3M$), 9 A.M. Monday ($9M$), or give up on going to the concert and not try at all (N). After going to the box office, depending on the line he observes, Bert can join the line, return later, or give up.

C. If Bert goes to the box office Sunday at 9 A.M., he is certain that the line will be short. If he goes there at 6 P.M. Sunday, Bert believes that the probability of a short line is .5, of a medium line .3, and of a long line .2. If the line is short at 6 P.M. and he does not join it, but returns at 3 A.M. Monday, Bert feels that the chances are .6 that it will then be of medium length, .3 that it will be long, and .1 that it will be hopelessly long (HL) with no chance of getting a ticket and no point to joining the line. If it is medium length at 6 P.M., Bert thinks that the probability is .7 that it will be long at 3 A.M., and .3 that it will be hopelessly long. If the line is long at 6 P.M., it is sure to be hopelessly long at 3 A.M. If at 3 A.M. the line is of medium length, Bert believes that the probability is .8 that it will be long at 9 A.M. and .2 that it will be hopelessly long. If the line is long at 3 A.M., it is sure to be hopelessly long at 9 A.M. (We assume that no event given zero chance of occurrence by Bert will occur and any event considered certain by Bert will occur.)

D. The *worst* possible outcome (designated W) that Bert can imagine, given his possible decisions, would be to go at 6 P.M. Sunday, find a medium line, join it and wait all night and then not get a ticket. The *best* possibility Bert can conceive (called B) is to go at 9 A.M. Monday, join a long line, and get a ticket. Bert's feelings about other possible outcomes lie between these extremes. The intermediate outcomes are evaluated by Bert, for reasons explained in the Raiffa reference, in terms of a lottery involving only chances at the worst outcome W and best outcome B. For example, if Bert feels that he would be indifferent between a lottery with a 20% chance at W and 80% at B and the guaranteed outcome "go at 3 A.M., find and join a medium length line, and get a ticket," he assigns the latter outcome the value .8, the probability of B in the lottery that would make him indifferent. (Hence, the worst outcome has value 0 and the best has value 1.) These values are assigned by Bert,

purely subjectively, based on his enthusiasm for hard rock, desire for sleep, distaste for trips to the box office, discomfort resulting from the uncertainty associated with waiting in a long line, etc.

E. Bert makes the Markovian assumption (see Chapter 13) that the probabilities for the length of the line depend only on the line length at the previous significant point in time (specifically, 6 P.M. Sunday or 3 A.M. Monday) and not on events at earlier times. Hence, Bert calculates, based on figures given in B above, that if he goes to the box office for the first time at 3 A.M. Monday, his chances of finding a medium line are $(.5)(.6)$ or $.3$, of finding a long line $(.5)(.3) + (.3)(.7)$ or $.36$, and of finding a hopelessly long line $(.5)(.1) + (.3)(.3) + .2$ or $.34$. If he goes first at 9 A.M. Monday his chance of finding a long line is $(.3)(.8) = .24$ and of finding a hopelessly long line is $.76$. If he goes at 6 P.M. Sunday and finds a short line, the chance of a long line at 9 A.M. Monday is $(.6)(.7)$ or $.42$, while the chance of a hopelessly long line is $.58$.

All of the above, and Bert's values for each decision-outcome sequence, are summarized in Figure 15.2, which shows what is called a decision tree. Squares represent decision points and lines coming from such points represent possible decisions. Circles represent chance events and each line emanating from a chance event represents a possible outcome. The number on a possible outcome line is Bert's assessment of its probability, given the decisions and outcomes represented by the sequence of lines leading to the line. The letters designating decisions and outcomes are defined in A–E above. The number in a triangle at the end of each path is Bert's subjective assessment of the value of the path, determined by Bert's preference for that path as compared to a lottery involving only the best and worst paths, as explained in D above.

Certain possible decisions are not included in the tree representation because they are clearly nonoptimal. For example, if Bert goes to the box office at 9 A.M. Monday and finds a long, but not hopelessly long, line, he should join the line. Giving up without trying at this point is foolish since Bert cannot hope to find a shorter line at 9 A.M. Monday, so rather than go to the box office and then give up, he would be better off giving up without even going. Any time Bert goes to the box office and finds the shortest possible line he obviously should never consider giving up, although before 9 A.M. Monday he does want to consider returning later as well as joining the line. If Bert chooses to go to the box office at 9 A.M. Sunday, he should definitely then join the short line. Otherwise there would be no point in going then since he knows that the line will be short.

Numerical solution of this problem now proceeds by the usual backward dynamic-programming method. Calling this an eight-stage problem because the most elaborate possible sequence of decisions and
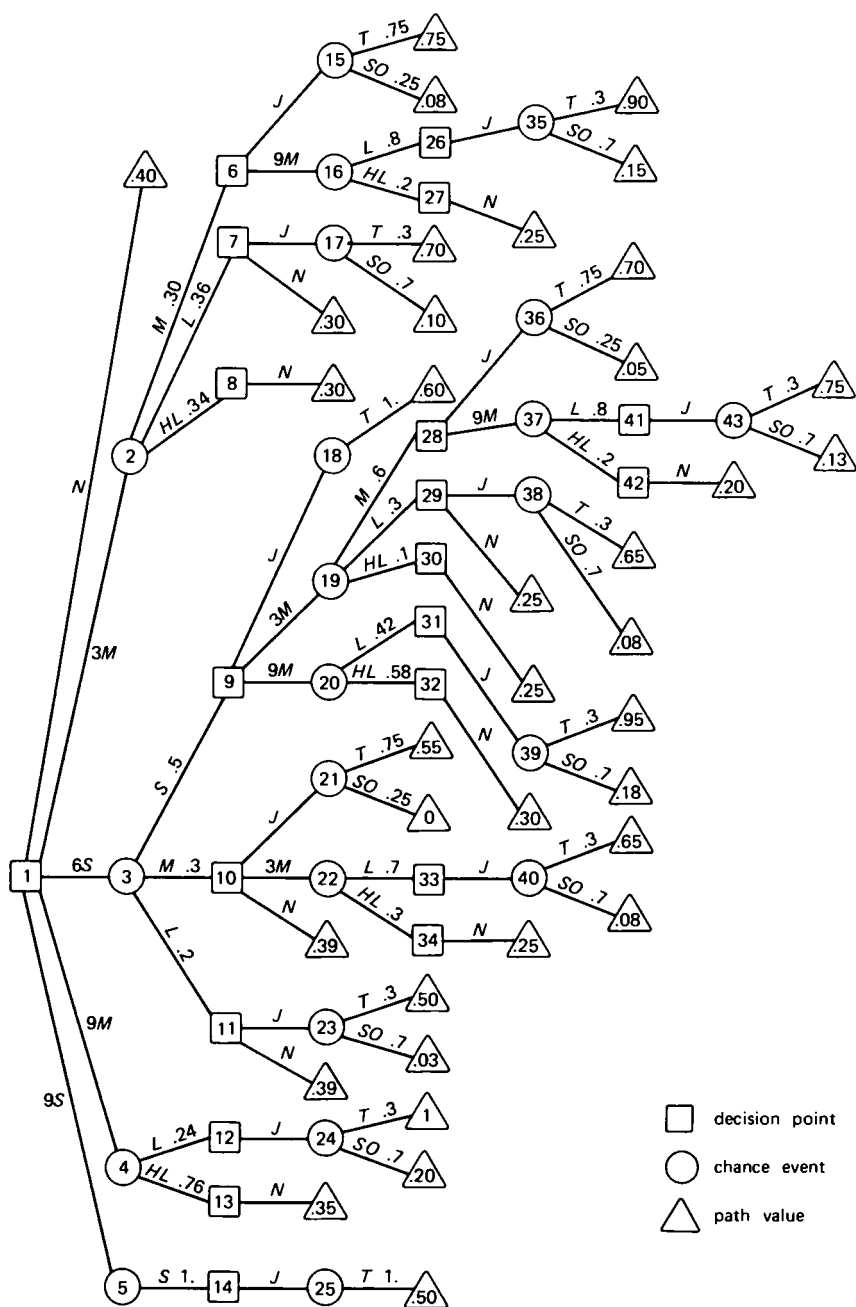
Figure 15.2

Table 15.2 · *Numerical solution of the rock concert problem*

Stage 8		Stage 7			Stage 6		Stage 5		
State	Value	State	Value		State	Value	State	Value	(Opt. dec.)
43	.316	42	.200		40	.251	34	.250	
		41	.316		39	.411	33	.251	
					38	.251	32	.300	
					37	.293	31	.411	
					36	.538	30	.250	
					35	.375	29	.251	(J)
							28	.538	(J)
							27	.250	
							26	.375	

Stage 4		Stage 3			Stage 2		Stage 1		
State	Value	State	Value	(Opt. dec.)	State	Value	State	Value	(Opt. dec.)
25	.500	14	.500		5	.500	1	.502	(6S)
24	.440	13	.350		4	.372			
23	.171	12	.440		3	.502			
22	.251	11	.390	(N)	2	.385			
21	.413	10	.413	(J)					
20	.347	9	.600	(J)					
19	.423	8	.300						
18	.600	7	.300	(N)					
17	.280	6	.583	(J)					
16	.350								
15	.583								

outcomes entails four decision stages and four outcome stages, we start at stage 8 and work backward. At each random outcome point, the value to Bert of starting at that point is the sum of terms, each one being the product of the probability of an outcome times the value of the situation after the occurrence of the outcome. At each decision point, the optimal decision is the one leading to the point with highest value. In Table 15.2 we show the values and optimal decisions that result from this calculation, where the nodes are numbered as in Figure 15.2. The value $S(19)$ of starting at the random event node 19, for example, is computed by

$$S(19) = (.6)S(28) + (.3)S(29) + (.1)S(30)$$
$$= (.6)(.650) + (.3)(.251) + (.1)(.250) = .490,$$

and the value at the decision node $S(1)$ is given by

$$S(1) = \max(.400, \, S(2), \, S(3), \, S(4), \, S(5))$$
$$= \max(.400, \, .385, \, .502, \, .372, \, .500)$$
$$= .502(6S).$$

The calculation tells us that Bert's optimal strategy is: Go to the box office at 6 P.M. Sunday and if the line is either short or of medium length, join it. If the line is long, give up on going to the concert. Bert feels as good

about pursuing this strategy as he would about playing a lottery with probability .502 of the best result B and .498 of the worst result W. Almost as attractive to Bert is joining the line at 9 A.M. Sunday, which entails a 24-hour wait but yields a sure ticket.

This formulation of the problem is an example of learning without the explicit use of Bayes' law. While Bert learns from inspecting the line, he does not explicitly view the situation in quite the same way as the previous two sections. If he did, he would say that a parameter (perhaps the D.P.'s local popularity) is fixed but unknown to him. He would assess the prior density of this parameter and would explain how he felt the chance event, the box-office line's length, would be influenced by this parameter. Then, having observed the line's length he would adjust the density of the parameter and he would use the new density to compute probabilities of future lengths of the line. The appropriate choice of the prior density and of its effect on line length would reproduce the probabilities that, in our above analysis, Bert gave us directly.

Another not completely typical property of this overly simplified analysis is Bert's ability to assess and evaluate all possible sequences of choices and outcomes. For more realistic problems, with more alternatives and possible outcomes, the decision analyst typically seeks to identify attributes that characterize possible situations. Then he seeks, for each attribute, the decision maker's subjective feeling about different scores for the attribute and also the decision maker's feelings about the relative importance of the attributes. From this information, the decision analyst attempts to synthesize a formula giving the decision maker's overall preferences for total situations. For example, in the above rock concert problem, the attributes might be (1) the number of daytime trips to the box office, (2) the number of nighttime trips to the box office, (3) the number of daytime hours waiting in line, (4) the number of nighttime hours waiting in line, and (5) possession of a ticket. The decision analyst would seek Bert's feelings about various scores on these attributes and also the relative importance to Bert of these scores. Then, given a bit more information about Bert's assessment of line-length probabilities, the decision analyst, using a computer, could find Bert's best strategy while allowing far more alternatives than we did in our example.

This synthesis of a person's preferences for whole real-world situations by means of calculations involving separate context-free component elements is a debatable practice, questioned by some contemporary philosophers and psychologists. They would argue that in certain areas where experience plays an important role in performance (such as chess or executive-level business policy planning), with experience comes a restructuring of the problem. While the problem may be seen by *novices* as involving the application of context-free rules or principles to weighted

attribute scores (for example, beginners in chess are taught to assign to each piece a numerical value), the *experienced decision maker* develops mental images (called paradigms) of different types of situations and develops preferences for those paradigm situations. These images or categories are holistic and not analyzable in terms of context-free components. (In chess, such categories are, for example, cramped position or out-of-control.) The decision maker develops, with experience, a sense of which paradigm best characterizes the confronted or contemplated real-world situation, and his preference stems from his preference for the paradigm situation.

Problem 15.8. Do a decision analysis of your life, or some subset thereof (like whether to stay in school, or whether to get up in the morning). Are your preferences for situations computable in terms of scores on attributes? The solution will not be found in the back of this book.

6. A Linear Dynamics, Quadratic Criterion Problem with Learning

In this rather mathematical last section we return to the model of Chapters 6 and 14, where we dealt with problems involving a linear dynamical system and a quadratic criterion. We assume now that the actual state of the system is not known with certainty, but that we learn about it as the process evolves by means of error-corrupted observations. (See Problem 6.11.) Furthermore, unlike Problem 6.11, we are asked to control the system while we learn about it.

Our model, which is general enough to allow us to develop the important ideas, but is not the most general model that can be treated by our procedures, is as follows.

The system obeys the dynamical rule

$$x(i + 1) = gx(i) + hy(i) + r(i) \qquad (i = 0, \dots, N - 1), \qquad (15.14)$$

where $x(i)$ is the state, $y(i)$ is the decision, and $r(i)$ is a normally distributed random variable, with $r(i)$ independent of $r(j)$ for $j \neq i$. For all i, the mean of $r(i)$ is $\bar{r}$ and the variance is σ^2. We do not know the true value of $x(i)$. We do know that $x(0)$ is chosen by sampling a normal distribution with known mean $\bar{x}(0)$ and variance $\sigma^2_{x(0)}$. Furthermore, at each stage i we observe the value of a variable $z(i)$ that is the sum of $x(i)$ and a random error term $u(i)$, where $u(i)$ is a normally distributed random variable, independent of $u(j)$ for $i \neq j$, with mean 0 and variance σ^2_u. The observation process can therefore be described by the equation

$$z(i) = x(i) + u(i). \qquad (15.15)$$

Based on the prior density of $x(0)$, on the observations $z(k)$ for $k = 0, 1, \dots, i$ and on the prior decisions $y(k)$ for $k = 0, 1, \dots, i - 1$, we are

to choose $y(i)$. We wish to do so in such a way as to minimize the expected total cost, where the cost J is given by

$$J = \sum_{i=0}^{N-1} \left(ax^2(i) + cy^2(i) \right) + lx^2(N). \tag{15.16}$$

We now assign the reader certain calculations required in the solution to this problem.

Problem 15.9. Let w be a normally distributed random variable, with mean μ and variance σ^2. Hence the density function for w, $p(w)$, is given by

$$p(w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(w - \mu)^2}{2\sigma^2} \right).$$

Given $p(w)$ and given y, let the random variable $w(i + 1)$ be given by

$$w(i + 1) = gw + hy + r,$$

where the normally distributed random variable r has density

$$p(r) = \frac{1}{\sqrt{2\pi\sigma_r^2}} \exp\left(-\frac{(r - \bar{r})^2}{2\sigma_r^2} \right).$$

What is the density function $p(w(i + 1)|y)$, the density of $w(i + 1)$ given y?

Problem 15.10. Given $p(w(i + 1)|y)$, let z be observed, where

$$z = w(i + 1) + u$$

and where the normally distributed random variable u has density

$$p(u) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left(-\frac{u^2}{2\sigma_u^2} \right).$$

Use Bayes' law to find $p(w(i + 1)|y, z)$, the density function of $w(i + 1)$, given z as well as y.

We now write the dynamic-programming recurrence relation for the problem given by (15.14)–(15.16). The state of the system at stage i is the information needed to compute the current density of the true state x, and this information is the present and all past observations $z(k)$, $k = 0, \ldots, i$, and all past decisions $y(k)$, $k = 0, \ldots, i - 1$. However, the state density is completely described by its mean and variance since Problem 15.10 has shown it to be normally distributed. The mean depends on all past observations and decisions and sums up all the information that can be deduced from them. Hence, we define the optimal expected value function by

$$S_i(\bar{w}, \sigma_w^2) = \text{the minimum remaining cost expected by us if}$$
the process starts at stage i with $z(i)$ having been observed and with our density for the true state (based on $z(i)$ and all previous z's and y's) being normal with mean $\bar{w}$ and variance σ_w^2. $\tag{15.17}$

Then

$$S_i(\overline{w}, \sigma_w^2) = \min_{\substack{y(i) \\ z(i+1)}} \mathop{E}_{x(i)} \left[ax^2(i) + cy^2(i) + S_{i+1}(\overline{w}(i+1), \sigma_{w(i+1)}^2) \right],$$

(15.18)

where the random variables $x(i)$ and $z(i+1)$ are conditioned on $\overline{w}$ and σ_w^2 and where, by the solution of Problem 15.10,

$$\overline{w}(i+1) = \frac{(g\overline{w} + hy + \overline{r})\sigma_u^2 + z(i+1)(g^2\sigma_w^2 + \sigma_r^2)}{g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2}$$

$$= g\overline{w} + hy + \overline{r} + \frac{g^2\sigma_w^2 + \sigma_r^2}{g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2} \left[z(i+1) - (g\overline{w} + hy + \overline{r}) \right] \quad (15.19)$$

and

$$\sigma_{w(i+1)}^2 = \frac{\sigma_u^2(g^2\sigma_w^2 + \sigma_r^2)}{g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2}. \quad (15.20)$$

(Note that (15.19) is the same type of formula that we obtained when solving Problem 6.11.)

The boundary condition is

$$S_N(\overline{w}, \sigma_w^2) = \mathop{E}_{x(N)} lx^2(N) = l\overline{w}^2 + l\sigma_w^2. \quad (15.21)$$

We now make the inductive hypothesis that $S_{i+1}(\overline{w}, \sigma_w^2)$ is of the form

$$S_{i+1}(\overline{w}, \sigma_w^2) = p(i+1)\overline{w}^2 + q(i+1)\overline{w} + t(i+1), \quad (15.22)$$

where p, q, and t may depend on σ_w^2. By (15.20) note that σ_w^2, unlike $\overline{w}$, is independent of all observations and decisions.

In the analysis that follows we make the (unnecessary) simplifying assumption that $\overline{r} = 0$. This makes the problem symmetric about $\overline{w} = 0$ and means that $q(i)$ is identically zero. Substituting assumption (15.22) into (15.18) we get, where $\overline{w}(i+1)$ is given by (15.19),

$$S_i(\overline{w}, \sigma_w^2) = \min_{\substack{y(i) \\ z(i+1)}} \mathop{E}_{x(i)} \left[ax^2(i) + cy^2(i) + p(i+1)\overline{w}^2(i+1) + t(i+1) \right]$$

$$= \min_y \mathop{E}_{z(i+1)} \left[a\overline{w}^2 + a\sigma_w^2 + cy^2 + p(i+1) \right.$$

$$\times \frac{(g\overline{w} + hy)^2\sigma_u^4 + 2z(i+1)(g^2\sigma_w^2 + \sigma_r^2)(g\overline{w} + hy)\sigma_u^2}{(g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2)^2}$$

$$+ p(i+1) \frac{z^2(i+1)(g^2\sigma_w^2 + \sigma_r^2)^2}{(g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2)^2} + t(i+1) \Bigg].$$

Now,

$$E\big(z(i+1)|\bar{w},\,\sigma_w^2\big) = g\bar{w} + hy$$

and

$$E\big(z^2(i+1)|\bar{w},\,\sigma_w^2\big) = (g\bar{w}+hy)^2 + g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2.$$

Hence

$$S_i\big(\bar{w},\,\sigma_w^2\big) = \min_y \Bigg[a\bar{w}^2 + a\sigma_w^2 + cy^2 + p(i+1)$$

$$\times \frac{(g\bar{w}+hy)^2\sigma_u^4 + 2(g\bar{w}+hy)(g^2\sigma_w^2+\sigma_r^2)(g\bar{w}+hy)\sigma_u^2}{(g^2\sigma_w^2+\sigma_r^2+\sigma_u^2)^2}$$

$$+ p(i+1)\frac{\big[(g\bar{w}+hy)^2 + g^2\sigma_w^2 + \sigma_r^2 + \sigma_u^2\big](g^2\sigma_w^2+\sigma_r^2)^2}{(g^2\sigma_w^2+\sigma_r^2+\sigma_u^2)^2} + t(i+1)\Bigg]$$

$$= \min_y \Bigg[a\bar{w}^2 + cy^2 + p(i+1)$$

$$\times \frac{(g\bar{w}+hy)^2\big[\sigma_u^4 + 2(g^2\sigma_w^2+\sigma_r^2)\sigma_u^2 + (g^2\sigma_w^2+\sigma_r^2)^2\big]}{(g^2\sigma_w^2+\sigma_r^2+\sigma_u^2)^2}$$

$$+ \text{terms independent of } \bar{w} \text{ and } y \Bigg].$$

The denominator cancels the last term of the numerator of the fraction on the right, so

$$S_i\big(\bar{w},\,\sigma_w^2\big) = \min_y \Big[a\bar{w}^2 + cy^2 + p(i+1)(g\bar{w}+hy)^2$$

$$+ \text{terms independent of } \bar{w} \text{ and } y \Big]. \tag{15.23}$$

The minimizing y satisfies

$$y(i) = -\frac{p(i+1)gh\bar{w}}{c+p(i+1)h^2} \tag{15.24}$$

so, substituting this y into (15.23),

$$S_i\big(\bar{w},\,\sigma_w^2\big) = \left[a + \frac{cp(i+1)g^2}{c+p(i+1)h^2} \right]\bar{w}^2 + \text{terms independent of } \bar{w}.$$

In view of (15.21), the inductive hypothesis is confirmed, and $p(i)$ satisfies

the recurrence relation

$$p(i) = a + \frac{cp(i+1)g^2}{c + p(i+1)h^2} , \qquad p(N) = l. \qquad (15.25)$$

Hence p is independent of σ_w^2 and also of the observations z and decisions y.

Interpreting the above results, we should begin solution by computing $p(i)$, $i = 0, \ldots, N$, by (15.25). We start at stage 0, prior to any observation, with $\bar{w} = \bar{x}(0)$ and $\sigma_w^2 = \sigma_{x(0)}^2$ since we are given that the prior density of the state is $N(\bar{x}(0), \sigma_{x(0)}^2)$. After observing $z(0)$, Bayes' law says that the posterior density of the state is

$$N\left(\frac{\bar{x}(0)\sigma_u^2 + z\sigma_{x(0)}^2}{\sigma_{x(0)}^2 + \sigma_u^2} , \frac{\sigma_{x(0)}^2 \sigma_u^2}{\sigma_{x(0)}^2 + \sigma_u^2} \right). \qquad (15.26)$$

This result follows from Problem 15.10 with $g = 1$, $h = 0$, $\bar{r} = 0$, $\sigma_r^2 = 0$. The mean of the normal distribution (15.26) is the value of $\bar{w}$ to be used in (15.24) to determine $y(0)$. After $z(1)$ has been observed $\bar{w}(1)$ is computed by (15.19). This value of $\bar{w}$ is then used in (15.24) to find $y(1)$ and this process continues until termination at stage N.

The above can be summarized by saying that at each stage we compute the mean of the state density based on what we have observed and our previous decisions, and then we use (15.24) with p given by (15.25) to determine the decision. Comparison with results (6.15) and (6.17) for the deterministic linear dynamics, quadratic criterion problem tells us that formulas (15.25) and (15.24) for p and y are the same except that $\bar{w}$, the expected state, replaces x, the true state. Hence, after determining the mean of the state density we choose y as if the mean were indeed the true state. The fact that our uncertainty about the state does not affect the decision is not to be expected and occurs only for problems with linear dynamics, quadratic criteria, and normally distributed random variables. In general, the reader should be warned *against* dealing with uncertainty by estimating whatever is unknown and then making the decision that is optimal based on the assumption that the estimate is the true value.

PROBLEM SOLUTIONS

Chapter 1

1.1. If we reason as in the chapter, we see that we can easily solve the problem for any given starting point if we know the value (cost) of the minimum-sum path from the two subsequent vertices to which the two different initial decisions lead. This means that the optimal value function and its arguments can be defined just as in (1.1) of the text. Likewise, recurrence relation (1.2) still correctly relates neighboring values. Now, however, there are five acceptable terminal points. Each has the property that the cost of getting *from* such a point *to* such a point is trivially zero, so the correct boundary condition is

$$S(4, y) = 0 \qquad \text{for all five acceptable terminal points}$$

or, explicitly,

$$S(4, 4) = 0, \quad S(4, 2) = 0, \quad S(4, 0) = 0, \quad S(4, -2) = 0, \quad S(4, -4) = 0.$$

Using arrows, rather than the P-table used in the text, to represent optimal decisions at each vertex, and numbers to denote the optimal value function at each vertex, the numerical solution, which is obtained by working backward from line B using (1.2), is shown in Figure S1.1. The optimal path is $(0, 0)$, $(1, 1)$, $(2, 2)$, $(3, 1)$, $(4, 2)$.

1.2. Again only the boundary condition is affected by the new stipulation. Now we can say that going *from* a terminal point *to* that point has a negative cost (the rebate). Hence the boundary condition becomes

$$S(4, 4) = -2, \quad S(4, 2) = -1, \quad S(4, 0) = -3, \quad S(4, -2) = -4, \quad S(4, -4) = -3.$$

The solution is shown in Figure S1.2. The optimal path is $(0, 0)$, $(1, 1)$, $(2, 0)$, $(3, -1)$, $(4, -2)$.

1.3. Using dynamic programming and the boundary condition of Problem 1.1, there are $N + (N - 1) + (N - 2) + \cdots + 1$ vertices at each of which two additions and a comparison are required, yielding $N(N + 1)$ additions and $N(N + 1)/2$ comparisons. (The $2N$ additions at stage N are unnecessary since they all add 0 to arc costs, but on a computer avoiding them would require considerable additional programming.) For brute-force enumeration there are 2^N paths, each requiring $N - 1$ additions and each but one requiring one comparison, hence $(N - 1)2^N$ additions and $2^N - 1$ comparisons are needed. For $N = 20$, dynamic programming requires 420 additions and 210 comparisons while brute-force enumeration involves roughly 20 million additions and a million comparisons.
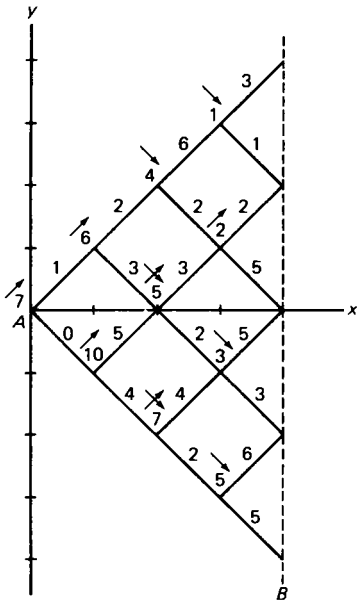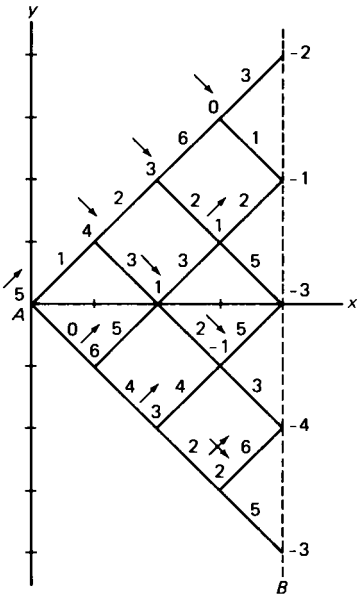
Figure S1.1

Figure S1.2

1.4. The solution is shown in Figure S1.3. The arrow at any particular vertex x points to the vertex that immediately precedes x on the optimal path to vertex x. Tracing back from B using the arrows, one obtains, naturally, the same path and cost as earlier. Just as with the earlier backward procedure, the solution requires 24 additions and 9 comparisons.
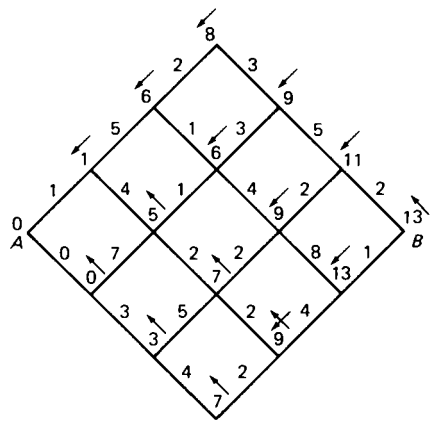
Figure S1.3

1.5. Define $S(x, y)$ as in (1.4) and use recurrence relation (1.5). The boundary condition is $S(0, 0) = 0$. After computing $S(4, y)$ for $y = 4, 2, 0, -2$, and -4, the minimum of these five numbers is the cost of the optimal path. Why? The solution is shown in Figure S1.4.

The backward procedure requires two additions and a comparison at each of 10 vertices, 20 additions, and 10 comparisons in all. The forward procedure requires two additions and one comparison at six vertices (which ones?), one addition at eight vertices, and also four comparisons to find the minimal $S(4, y)$, totaling to 20 additions and 10 comparisons, just as before.

1.6. Work backward from line B to obtain the cost of the best path from each vertex to line B, using the boundary condition $S(6, 2) = 2$, $S(6, 0) = 3$, $S(6, -2) = 1$. After determining
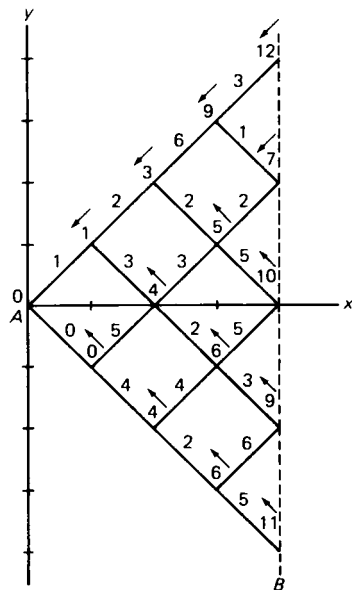

Figure S1.4

$S(0, 2)$, $S(0, 0)$, and $S(0, -2)$, add the initial-point cost and find the minimum of these three values. Its argument is the optimal starting point. (Why?) Use the arrows to trace out the optimal path. The solution is shown in Figure S1.5. The minimum $S(0, y)$ is $S(0, 2)$. The path is $(0, 2)$, $(1, 1)$, $(2, 2)$, $(3, 1)$, $(4, 0)$, $(5, 1)$, $(6, 2)$.

One could equally well have first computed forward from line A, letting $S(0, 2) = 2$, $S(0, 0) = 3$, $S(0, -2) = 2$, and then minimized $S(6, y)$, etc.
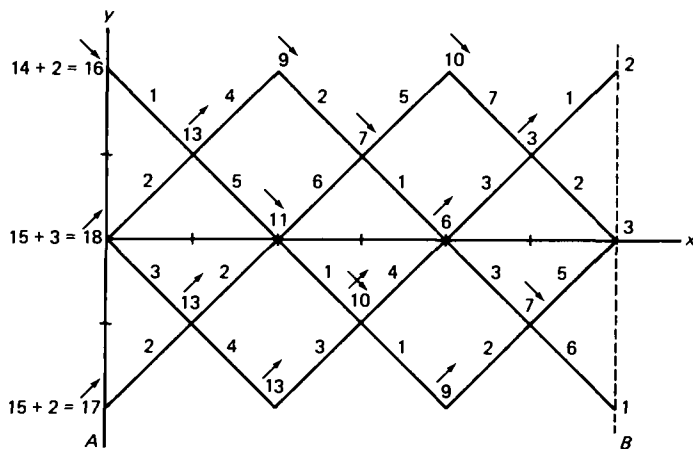

Figure S1.5

1.7. Define

> $S(x, y, z) =$ the minimum attainable sum of arc numbers plus turn penalties
> if we start at the vertex (x, y), having arrived at (x, y) by a move
> in the direction z, where $z = 0$ denotes diagonally upward and
> $z = 1$ denotes diagonally downward.

Then, by the principle of optimality,

$$S(x, y, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0) \\ 3 + a_d(x, y) + S(x + 1, y - 1, 1) \end{bmatrix},$$

$$S(x, y, 1) = \min \begin{bmatrix} 3 + a_u(x, y) + S(x + 1, y + 1, 0) \\ a_d(x, y) + S(x + 1, y - 1, 1) \end{bmatrix},$$

and the boundary conditions are

$$S(6, 0, 0) = 0, \qquad S(6, 0, 1) = 0.$$

The answer is the minimum of $S(0, 0, 0)$ and $S(0, 0, 1)$ or, equivalently,

$$\text{answer} = \min \begin{bmatrix} a_u(0, 0) + S(1, 1, 0) \\ a_d(0, 0) + S(1, -1, 1) \end{bmatrix}.$$

1.8. Formulas (1.9) and (1.10) each require two additions and a comparison. Hence four additions and two comparisons are required at each of $(N/2 + 1)^2$ vertices, assuming edge vertices are treated the same as the others. Consequently, roughly N^2 additions and $N^2/2$ comparisons are required. This is about twice the computation needed to solve the same problem without turn penalties.

1.9. Just as for the backward procedure of the text and of Problem 1.7, there are two equally acceptable solutions.

Solution 1. Define

$S(x, y, z) =$ the minimum attainable sum of arc numbers plus turn penalties if we start at $(0, 0)$ and go to (x, y), arriving at (x, y) be a move in direction z, where $z = 0$ denotes diagonally upward and $z = 1$ denotes diagonally downward.

By the principle of optimality

$$S(x, y, 0) = a_u(x - 1, y - 1) + \min\begin{bmatrix} S(x - 1, y - 1, 0) \\ 3 + S(x - 1, y - 1, 1) \end{bmatrix},$$

$$S(x, y, 1) = a_d(x - 1, y + 1) + \min\begin{bmatrix} 3 + S(x - 1, y + 1, 0) \\ S(x - 1, y + 1, 1) \end{bmatrix},$$

and

$$S(0, 0, 0) = 0, \qquad S(0, 0, 1) = 0.$$

The answer is the minimum of $S(6, 0, 0)$ and $S(6, 0, 1)$.

Solution 2. Define

$S(x, y, z) =$ the minimum attainable sum of arc numbers plus turn penalties if we start at $(0, 0)$, go to (x, y), and then leave (x, y) in direction z, where $z = 0$ denotes diagonally up and $z = 1$ denotes diagonally down. $S(x, y, z)$ does not include the cost of the agreed upon arc out of (x, y) but does include the turn cost.

Then,

$$S(x, y, 0) = \min\begin{bmatrix} a_u(x - 1, y - 1) + S(x - 1, y - 1, 0) \\ 3 + a_d(x - 1, y + 1) + S(x - 1, y + 1, 1) \end{bmatrix},$$

$$S(x, y, 1) = \min\begin{bmatrix} 3 + a_u(x - 1, y - 1) + S(x - 1, y - 1, 0) \\ a_d(x - 1, y + 1) + S(x - 1, y + 1, 1) \end{bmatrix},$$

and

$$S(0, 0, 0) = 0, \qquad S(0, 0, 1) = 0.$$

The answer is the minimum of $S(6, 0, 0)$ and $S(6, 0, 1)$.

There are other slightly different correct answers. For example, the arc out of the vertex (x, y) could be included in the cost $S(x, y, z)$.

1.10.

$$S(x, y, 0) = a_u(x, y) + \min\begin{bmatrix} S(x + 1, y + 1, 0) \\ S(x + 1, y + 1, 1) \end{bmatrix}.$$

$$S(x, y, 1) = a_d(x, y) + \min\begin{bmatrix} S(x + 1, y - 1, 0) \\ S(x + 1, y - 1, 1) \end{bmatrix}.$$

$$S(6, 0, 0) = 0, \qquad S(6, 0, 1) = 0.$$

This procedure requires roughly twice the computation of the efficient one given in the text, but it will yield the correct result.

1.11. Let $S(x, y, z, w) =$ the minimum cost from vertex (x, y) to the end, vertex $(N, 0)$, where z is defined as in Problem 1.7 and w is the number of direction changes prior to (x, y) (i.e., between $(0, 0)$ and (x, y)).

$$S(x, y, 0, w) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0, w) \\ w + 1 + a_d(x, y) + S(x + 1, y - 1, 1, w + 1) \end{bmatrix}.$$

$$S(x, y, 1, w) = \min \begin{bmatrix} w + 1 + a_u(x, y) + S(x + 1, y + 1, 0, w + 1) \\ a_d(x, y) + S(x + 1, y - 1, 1, w) \end{bmatrix}.$$

$$S(N, 0, 0, w) = S(N, 0, 1, w) = 0 \quad \text{for} \quad w = 1, \ldots, N - 1.$$

Answer is minimum of $S(0, 0, 0, 0)$ and $S(0, 0, 1, 0)$.

A forward procedure would define x, y, and z as in Solution 1 of Problem 1.9 and w as the number of direction changes prior to (x, y).

$S(x, y, z, w) =$ the minimum cost from $(0, 0)$ to (x, y) with z and w defined above.

$$S(x, y, 0, w) = \min \begin{bmatrix} w + a_u(x - 1, y - 1) + S(x - 1, y - 1, 1, w - 1) \\ a_u(x - 1, y - 1) + S(x - 1, y - 1, 0, w) \end{bmatrix}.$$

$$S(x, y, 1, w) = \min \begin{bmatrix} a_d(x - 1, y + 1) + S(x - 1, y + 1, 1, w) \\ w + a_d(x - 1, y + 1) + S(x - 1, y + 1, 0, w - 1) \end{bmatrix}.$$

$$S(0, 0, 0, 0) = S(0, 0, 1, 0) = 0.$$

Answer is

$$\min_{w = 1, \ldots, N - 1} \left[\min \left\{ \begin{matrix} S(N, 0, 0, w) \\ S(N, 0, 1, w) \end{matrix} \right\} \right].$$

There are other correct procedures. All involve four variables and an equivalent amount of computation.

1.12. $S(x, y, z, w) =$ the minimum cost from (x, y) to $(N, 0)$ with z defined as in Problem 1.7 and $w = 0$ if an even number of direction changes occurred between $(0, 0)$ and (x, y) and $w = 1$ if an odd number occurred.

$$S(x, y, 0, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0, 0) \\ a_d(x, y) + 3 + S(x + 1, y - 1, 1, 1) \end{bmatrix},$$

$$S(x, y, 0, 1) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0, 1) \\ a_d(x, y) + S(x + 1, y - 1, 1, 0) \end{bmatrix},$$

and similarly for $z = 1$.

$$S(N, 0, z, w) = 0 \quad \text{for} \quad z = 0, 1 \quad \text{and} \quad w = 0, 1.$$

Answer is minimum of $S(0, 0, 0, 0)$ and $S(0, 0, 1, 0)$.

There are other, similar, correct backward and forward procedures.

1.13. $S(x, y, z, w)$ is defined as in Problem 1.11.

$$S(x, y, 0, w) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0, w) \\ a_d(x, y) + S(x + 1, y - 1, 1, w + 1) \end{bmatrix},$$

and similarly for $S(x, y, 1, w)$. S is computed for $w = 0, 1, \ldots, m$. If $w = m$, do not consider the decision that changes direction. This can be accomplished by defining $S(x, y, z, m + 1) = \infty$.

$$S(N, 0, z, w) = 0 \quad \text{for} \quad z = 0, 1 \quad \text{and} \quad w = 0, 1, \ldots, m.$$

Answer is minimum of $S(0, 0, 0, 0)$ and $S(0, 0, 1, 0)$.

1.14. Imagine that you start with one "free arc" coupon, to be used whenever you choose. $S(x, y, z) =$ the minimum cost from (x, y) to $(N, 0)$ starting with z coupons, $z = 0$ or 1.

$$S(x, y, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0) \\ a_d(x, y) + S(x + 1, y - 1, 0) \end{bmatrix}.$$

$$S(x, y, 1) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 1) \\ S(x + 1, y + 1, 0) \\ a_d(x, y) + S(x + 1, y - 1, 1) \\ S(x + 1, y - 1, 0) \end{bmatrix}.$$

$$S(N, 0, 0) = S(N, 0, 1) = 0.$$

Answer is $S(0, 0, 1)$.

1.15. $S(x, y) =$ the minimum cost from (x, y) to $(N, 0)$.

$$S(x, y) = \min \begin{bmatrix} \max\{a_u(x, y), S(x + 1, y + 1)\} \\ \max\{a_d(x, y), S(x + 1, y - 1)\} \end{bmatrix}.$$

$$S(N, 0) = 0.$$

Answer is $S(0, 0)$.

Note the maximum inside the brackets. The cost of an up decision is the larger of the encountered arc and $S(x + 1, y + 1)$, the smallest possible largest arc that need be encountered between $(x + 1, y + 1)$ and the end.

Problems involving minimizing the worst cost occur frequently. In inventory problems, the storage facility must be as large as the largest inventory level. In control problems, one often wishes to minimize the largest deviation from a desired trajectory.

1.16. The problem combines ideas from Problems 1.14 and 1.15. Define $S(x, y, z) =$ the minimum cost starting from (x, y) where we start with z "free arc" coupons, $z = 0$ or 1.

$$S(x, y, 0) = \min \begin{bmatrix} \max\{a_u(x, y), S(x + 1, y + 1, 0)\} \\ \max\{a_d(x, y), S(x + 1, y - 1, 0)\} \end{bmatrix}.$$

$$S(x, y, 1) = \min \begin{bmatrix} \max\{a_u(x, y), S(x + 1, y + 1, 1)\} \\ S(x + 1, y + 1, 0) \\ \max\{a_d(x, y), S(x + 1, y - 1, 1)\} \\ S(x + 1, y - 1, 0) \end{bmatrix}.$$

$$S(N, 0, 0) = S(N, 0, 1) = 0.$$

Answer is $S(0, 0, 1)$.

1.17. $S(x, y, z) =$ the minimum cost starting from (x, y) with z coupons on hand.

$$S(x, y, z) = \min \begin{bmatrix} \min_{w = 0, 1, \ldots, z} \{a_u(x, y) - g_u(x, y, w) + S(x + 1, y + 1, z - w)\} \\ \min_{w = 0, 1, \ldots, z} \{a_d(x, y) - g_d(x, y, w) + S(x + 1, y - 1, z - w)\} \end{bmatrix}.$$

$$S(N, 0, z) = 0 \qquad \text{for} \quad z = 0, 1, \ldots, Z.$$

Answer is $S(0, 0, Z)$.

1.18. *Solution 1.* Compute $S(x, y)$ by (1.1), (1.2), (1.3) (with six replaced by N). Answer is $\min_R[p(x, y) + S(x, y)]$, where R is the set of all vertices (x, y) of the network.

Solution 2. Define $S(x, y)$ as the minimum cost from any initial vertex to (x, y).

$$S(x, y) = \min \begin{bmatrix} p(x, y) \\ a_d(x - 1, y + 1) + S(x - 1, y + 1) \\ a_u(x - 1, y - 1) + S(x - 1, y - 1) \end{bmatrix}.$$

$$S(0, 0) = p(0, 0).$$

Answer is $S(N, 0)$. When tracing the optimal path back from $(N, 0)$, whenever $p(x, y)$ yields the minimum, then (x, y) is the optimal starting vertex.

1.19. *Solution 1.* Define $S(x, y, z)$ as in Problem 1.7.

$$S(x, y, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0) \\ a_d(x, y) + a_d(x + 1, y - 1) + S(x + 2, y - 2, 1) \end{bmatrix}.$$

Note that we have here a situation in which the stage variable x increases by more than one for certain decisions. The backward computation is still feasible since S at $x + 2$ is known when S at x is to be computed.

A similar formula holds for $S(x, y, 1)$. $S(N, 0, 0) = S(N, 0, 1) = 0$. Answer is the minimum of $S(0, 0, 0)$ and $S(0, 0, 1)$.

Solution 2. Define $S(x, y, z, w) =$ the minimum cost from (x, y) to $(N, 0)$ where z is as in Problem 1.7 and $w = 0$ means a direction change is allowed at the vertex and $w = 1$ means a direction change is not allowed.

$$S(x, y, 0, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0, 0) \\ a_d(x, y) + S(x + 1, y - 1, 1, 1) \end{bmatrix}.$$

Similarly for $S(x, y, 1, 0)$.

$$S(x, y, 0, 1) = a_u(x, y) + S(x + 1, y + 1, 0, 0).$$

Similarly for $S(x, y, 1, 1)$.

$$S(N, 0, z, w) = 0 \quad \text{for} \quad z = 0, 1 \quad \text{and} \quad w = 0, 1.$$

Answer is minimum of $S(0, 0, 0, 0)$ and $S(0, 0, 1, 0)$.

1.20. $S(x, y, z) =$ the minimum-cost path from (x, y) to $(N, 0)$ where the sum of the arc costs from $(0, 0)$ to (x, y) is z.

$$S(x, y, z) = \min \begin{bmatrix} \max\{a_u(x, y), S(x + 1, y + 1, z + a_u(x, y))\} \\ \max\{a_d(x, y), S(x + 1, y - 1, z + a_d(x, y))\} \end{bmatrix}.$$

$$S(x, y, z) = \infty \quad \text{for} \quad z > Z.$$
$$S(N, 0, z) = 0 \quad \text{for} \quad z \le Z.$$

Answer is $S(0, 0, 0)$.

Computation of $S(x, y, z)$ for $z = 0, 1, \ldots, Z$ will yield the correct result. A range of z values from a to b, $a > 0$, $b < Z$ will be correct if no path from $(0, 0)$ to (x, y) can have sum $< a$ or $> b$.

1.21. Define $S(x, y)$ by (1.1).

$$S(x, y) = \min \begin{bmatrix} a_u(x, y) \cdot S(x + 1, y + 1) \\ a_d(x, y) \cdot S(x + 1, y - 1) \end{bmatrix}.$$

$$S(N, 0) = 1.$$

Answer is $S(0, 0)$. Note that products can be handled by dynamic programming as easily as sums. Also note the boundary condition used for products.

The problem can be converted to a summation criterion by noting that minimizing the product is equivalent to minimizing the logarithm of the product (if the product is positive), which equals the sum of the logarithms of the individual costs.

1.22. We do not know whether to maximize or minimize the remaining product unless we know whether the product of costs from $(0, 0)$ to (x, y) is positive or negative. Hence, define $S(x, y, z) =$ the minimum-cost path from (x, y) to $(N, 0)$ when $z = 0$ and the maximum-cost path from (x, y) to $(N, 0)$ when $z = 1$.

$$S(x, y, 0) = \min \begin{bmatrix} \{a_u(x, y)\cdot S(x + 1, y + 1, 0)\} & \text{if} & a_u(x, y) \geqslant 0, & \text{or} \\ \{a_u(x, y)\cdot S(x + 1, y + 1, 1)\} & \text{if} & a_u(x, y) < 0. \\ \{a_d(x, y)\cdot S(x + 1, y - 1, 0)\} & \text{if} & a_d(x, y) \geqslant 0, & \text{or} \\ \{a_d(x, y)\cdot S(x + 1, y - 1, 1)\} & \text{if} & a_d(x, y) < 0. \end{bmatrix}.$$

$$S(x, y, 1) = \max \begin{bmatrix} \{a_u(x, y)\cdot S(x + 1, y + 1, 1)\} & \text{if} & a_u(x, y) \geqslant 0, & \text{or} \\ \{a_u(x, y)\cdot S(x + 1, y + 1, 0)\} & \text{if} & a_u(x, y) < 0. \\ \{a_d(x, y)\cdot S(x + 1, y - 1, 1)\} & \text{if} & a_d(x, y) \geqslant 0, & \text{or} \\ \{a_d(x, y)\cdot S(x + 1, y - 1, 0)\} & \text{if} & a_d(x, y) < 0. \end{bmatrix}.$$

$$S(N, 0, 0) = S(N, 0, 1) = 1.$$

Answer is $S(0, 0, 0)$.

1.23. Computing $S(y_1, y_2, 2)$ requires one iteration, then $S(y_1, y_2, 4)$ requires a second, $S(y_1, y_2, 8)$ a third, $S(y_1, y_2, 16)$ a fourth, then combining $S(y_1, y_2, 8)$ and $S(y_1, y_2, 16)$ yields $S(y_1, y_2, 24)$ at the fifth iteration, $S(y_1, y_2, 24)$ and $S(y_1, y_2, 2)$ yield $S(y_1, y_2, 26)$ on the sixth iteration, and we get $S(y_1, y_2, 27)$ on the *seventh* iteration.

A better procedure would be to get $S(y_1, y_2, 2)$ on the first iteration, then $S(y_1, y_2, 3)$ using $S(y_1, y_2, 2)$ and the boundary condition $S(y_1, y_2, 1)$ on the second. Then combining $S(y_1, y_2, 3)$ with itself gives $S(y_1, y_2, 6)$ on the third and again doubling-up gives $S(y_1, y_2, 12)$ on the fourth and $S(y_1, y_2, 24)$ on the fifth iteration. Combining $S(y_1, y_2, 24)$ with $S(y_1, y_2, 3)$ gives $S(y_1, y_2, 27)$, the answer, on the *sixth* iteration, improving by one on the first method above.

How to find systematically the procedure that minimizes the number of iterations for general N is at present an unsolved problem.

Chapter 2

2.1. The optimal value function is defined by

$S(x, k) =$ the minimum cost of owning a machine from year k through N,
        starting year k with a machine just turned age x.

The recurrence relation is

$$S(x, k) = \min \begin{bmatrix} \text{Buy:} & p - t(x) + c(0) + S(1, k + 1) \\ \text{Keep:} & c(x) + S(x + 1, k + 1) \end{bmatrix}. \qquad (S2.1)$$

The boundary condition is

$$S(x, N + 1) = -s(x). \qquad (S2.2)$$

The recurrence relation (S2.1) evaluates, first, the "buy decision" by adding the cost during year k of buying a new machine (the purchase price plus the year's operating cost less the trade-in value) to the minimum attainable cost for the remaining process, which starts year $k + 1$ with a one-year-old machine. Then it evaluates the decision "keep the current machine for at least one more year" by adding the cost of operating the old machine for a year to the minimum cost of the remaining process, which in this case starts year $k + 1$ with a machine of age $x + 1$ (one year older than it was at the start of year k). The better of these two alternatives is recorded as the value of $S(x, k)$ since the principle of optimality assures us that no nonoptimal continuation from year $k + 1$ to N need be considered.

One should also note the optimal decision in this situation, writing $P(x, k) = B$ (buy) if line 1 on the right of (S2.1) yields the minimum, and $P(x, k) = K$ (keep) if line 2 yields the minimum.

The boundary condition states that a rebate (negative cost) of $s(x)$ occurs if the machine has just turned age x at the start of year $N + 1$, i.e., at the end of year N.

Naturally, our use of the particular letters S, x, and k to denote certain quantities is quite arbitrary, and any symbols, if appropriately defined, are equally acceptable. An equivalent and equally acceptable boundary condition is

$$S(x, N) = \min\left[\begin{array}{l} p - t(x) + c(0) - s(1) \\ c(x) - s(x + 1) \end{array}\right].$$

If y denotes the age of the starting (incumbent) machine, (S2.2) must be tabulated for $x = 1, 2, \ldots, N$ and $y + N$ since various decisions during the first N years could result in each of these terminal ages. (Age $y + N$ corresponds to the prior decisions never to replace the incumbent machine.) Then $S(x, N)$ is computed, using (S2.1) for $x = 1, \ldots, N - 1$ and $y + N - 1$. (The incumbent machine, if never replaced, will be of age $y + N - 1$ starting year N.) Next compute $S(x, N - 1)$ for $x = 1, \ldots, N - 2$ and $y + N - 2$, etc. until $S(x, 2)$ is computed for $x = 1$ and $y + 1$. Finally compute $S(y, 1)$, the minimum cost for the given problem.

2.2. According to the boundary condition (S2.2)

$$S(1, 6) = -25, \quad S(2, 6) = -17, \quad S(3, 6) = -8, \quad S(4, 6) = S(5, 6) = S(7, 6) = 0.$$

Now, using (S2.1) to compute $S(x, 5)$:

$$S(1, 5) = \min\left[\begin{array}{l} 50 - 32 + 10 + S(1, 6) \\ 13 + S(2, 6) \end{array}\right] = -4, \qquad P(1, 5) = \text{keep};$$

$$S(2, 5) = \min\left[\begin{array}{l} 50 - 21 + 10 + S(1, 6) \\ 20 + S(3, 6) \end{array}\right] = 12, \qquad P(2, 5) = \text{keep};$$

$$S(3, 5) = \min\left[\begin{array}{l} 24 \\ 40 \end{array}\right] = 24, \qquad P(3, 5) = \text{buy};$$

$$S(4, 5) = \min\left[\begin{array}{l} 30 \\ 70 \end{array}\right] = 30, \qquad P(4, 5) = \text{buy};$$

$$S(6, 5) = \min\left[\begin{array}{l} 35 \\ 100 \end{array}\right] = 35, \qquad P(6, 5) = \text{buy}.$$

Next, use (S2.1) and the above results to determine $S(x, 4)$:

$$S(1, 4) = \min\left[\begin{array}{l} 28 + S(1, 5) \\ 13 + S(2, 5) \end{array}\right] = 24, \qquad P(1, 4) = \text{buy};$$

$$S(2, 4) = \min\left[\begin{array}{l} 35 \\ 44 \end{array}\right] = 35, \qquad P(2, 4) = \text{buy};$$

$$S(3, 4) = \min\left[\begin{array}{l} 45 \\ 70 \end{array}\right] = 45, \qquad P(3, 4) = \text{buy};$$

$$S(5, 4) = \min\left[\begin{array}{l} 56 \\ 135 \end{array}\right] = 56, \qquad P(5, 4) = \text{buy}.$$

Next, determine $S(x, 3)$:

$$S(1, 3) = \min\left[\begin{array}{l} 52 \\ 48 \end{array}\right] = 48, \qquad P(1, 3) = \text{keep};$$

$$S(2, 3) = \min\left[\begin{array}{l} 63 \\ 65 \end{array}\right] = 63, \qquad P(2, 3) = \text{buy};$$

$$S(4, 3) = \min\left[\begin{array}{l} 79 \\ 126 \end{array}\right] = 79, \qquad P(4, 3) = \text{buy}.$$

Next, determine $S(x, 2)$:

$$S(1, 2) = \min\begin{bmatrix} 76 \\ 76 \end{bmatrix} = 76, \qquad P(1, 2) = \text{buy or keep};$$

$$S(3, 2) = \min\begin{bmatrix} 97 \\ 119 \end{bmatrix} = 97, \qquad P(3, 2) = \text{buy}.$$

Finally,

$$S(2, 1) = \min\begin{bmatrix} 115 \\ 117 \end{bmatrix} = 115, \qquad P(2, 1) = \text{buy}.$$

The optimal sequence of decisions is: buy at year 1 since $P(2, 1) =$ buy; either buy or keep at the start of year 2 since $P(1, 2) =$ either; if we choose to buy, then keep during year 3 since $P(1, 3) =$ keep; then buy at the start of year 4 since $P(2, 4) =$ buy and keep during year 5 since $P(1, 5) =$ keep. If we choose to keep during year 2, then buy at the start of year 3, buy at the start of year 4, and keep during year 5. The two optimal policies are, therefore, *BBKBK* and *BKBBK*. We can add up the costs of these two policies to verify that they cost 115, the value we got for $S(2, 1)$. Cost$(BBKBK) = p - t(2) + c(0) + p - t(1) + c(0) + c(1) + p - t(2) + c(0) + c(1) - s(2) = 50 - 21 + 10 + 50 - 32 + 10 + 13 + 50 - 21 + 10 + 13 - 17 = 115$ and cost $(BKBBK) = 50 - 21 + 10 + 13 + 50 - 21 + 10 + 50 - 32 + 10 + 13 - 17 = 115$. This calculation does not prove that we have not made an error, but it does increase our confidence in the correctness of the solution.

2.3. Each evaluation of (S2.1) requires four additions and a comparison. Now imagine that you are actually performing the computation and count the number of values of S that you would need to calculate. At the start of year N, there are N values of S required ($x = 1, \ldots, N - 1$ and $y + N - 1$). Verify this for Problem 2.2, where $N = 5$ and $y = 2$. At the start of year $N - 1$, $N - 1$ values of S are needed, etc., until at the start of year 1, one value must be computed. Hence $\sum_{i=1}^{N} i = N(N + 1)/2$ values of S must be computed, so a total of $2N(N + 1)$ additions and $N(N + 1)/2$ comparisons are needed.

When we ask for the approximate number of calculations, we really want an idea of how the answer grows with N and how it depends on the range of values that the state variables can take on (which in this example depends on N, but generally is an independent choice in the model formulation). For this problem we would consider KN^2 with K any number between 1 and, say, 5 an appropriate answer for the total number of calculations. Hence the reader should feel free to be fairly approximate in counting additions or comparisons, or adding up the number of values of S required.

2.4. Define the optimal value function by

$S(x, k) =$ the minimum cost of getting to the situation that you own a machine just turned age x at the start of year k, given that you start year 1 with a machine of age y.

The recurrence relation is

$$S(1, k) = \min_{\{J\}} [p - t(j) + c(0) + S(j, k - 1)]$$

$$= p + c(0) + \min_{\{J\}} [-t(j) + S(j, k - 1)], \qquad (S2.3)$$

where $\{J\}$ denotes the set of values $j = 1, 2, \ldots, k - 2$ and $k + y - 2$. For $x = 2, \ldots, k - 1$ and $k + y - 1$,

$$S(x, k) = c(x - 1) + S(x - 1, k - 1). \qquad (S2.4)$$

We have used the convention that when $q < 1$, $x = 1, 2, \ldots, q$ means no such value of x
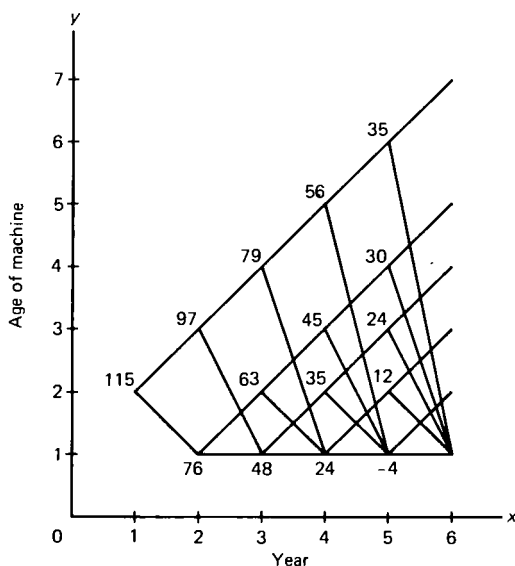
Figure S2.1

exists. When $q = 1$, the symbol means consider $x = 1$ only and when q equals, say 4, it means to use $x = 1, 2, 3, 4$.

The boundary condition is

$$S(y, 1) = 0. \tag{S2.5}$$

The optimal terminal age is equal to that value of z that yields the minimum of $-s(z) + S(z, N + 1)$, $z = 1, 2, \ldots, N$ and $N + y$. $\tag{S2.6}$

When computing $S(1, k)$, always note the value of j that yields the minimum since this is the age of the machine traded-in at the start of the previous year on the optimal path.

Equation (S2.4) is computed at a total of $1 + 2 + 3 + \cdots + N$ points, requiring $N(N + 1)/2$ additions. Recurrence relation (S2.3) requires two additions plus one addition and no comparisons when $k = 2$, plus two additions and one comparison when $k = 3$ and ultimately plus N additions and $N - 1$ comparisons at stage $N + 1$, totaling $2N + [N(N + 1)/2]$ additions and $(N - 1)N/2$ comparisons. The minimization (S2.6) requires a negligible $N + 1$ additions and N comparisons. This totals to approximately N^2 additions and $N^2/2$ comparisons, a bit less than for the backward procedure due to the special economy of removing $p + c(0)$ from the minimization in (S2.3), but the work remains of the same order of magnitude.

2.5. See Figure S2.1.

2.6. Define S as in Problem 2.1.

$$S(x, k) = \min_{j = 0, \ldots, M-1} [u(x, j) + c(j) + S(j + 1, k + 1)].$$

$$S(x, N + 1) = -s(x).$$

There are M possible values of x for each value of k, and $N - 1$ values of k (neglecting stage 1). Furthermore, each x, k pair requires $2M$ additions and $M - 1$ comparisons.

2.7. Define S as in Problem 2.1.

$$S(x, k) = \min_{j = 0, \ldots, M-1} [u_k(x, j) + c_k(j) + S(j + 1, k + 1)].$$

Many more data are needed to define this problem, but no more computation is required than for Problem 2.6.

2.8. Define

$S(x, k) =$ the minimum cost of the remaining process if you start year k with an owned machine of age x;

$T(k) =$ the minimum cost of the remaining process if you start year k without an owned machine, having leased one during the preceding year.

$$S(x, k) = \min \begin{bmatrix} \text{Buy:} & p - t(x) + c(0) + S(1, k + 1) \\ \text{Keep:} & c(x) + S(x + 1, k + 1) \\ \text{Lease:} & -s(x) + l + c(0) + T(k + 1) \end{bmatrix}.$$

$$T(k) = \min \begin{bmatrix} \text{Buy:} & p + c(0) + S(1, k + 1) \\ \text{Lease:} & l + c(0) + T(k + 1) \end{bmatrix}.$$

$$S(x, N + 1) = -s(x).$$
$$T(N + 1) = 0.$$

The answer is $S(0, 1)$.

As in Problem 2.3, roughly $N^2/2$ values of S must be computed, as well as a negligible $N - 1$ values of T. Each value of S requires seven additions and two comparisons, a total of nine computations, so about $9N^2/2$ computations are required overall.

2.9. A consultant would need to know the year, age of current machine, and age at trade of the previous machine. Let $S(x, k, z) =$ the minimum cost of the remaining process if you start year k with a machine of age x and you must trade on or before age z, where $z = \infty$ means the incumbent machine has never been replaced.

If $x < z$,

$$S(x, k, z) = \min \begin{bmatrix} p - t(x) + c(0) + S(1, k + 1, x) \\ c(x) + S(x + 1, k + 1, z) \end{bmatrix}. \tag{S2.7}$$

If $x = z$ you must trade, so

$$S(x, k, z) = p - t(x) + c(0) + S(1, k + 1, x).$$

Furthermore,

$$S(x, N + 1, z) = -s(x).$$

The function S is computed for x in the same range as in Problem 2.1. Possible values of z depend on k and x in a complicated way. If S is computed for more situations than can actually occur, the answer will still be correct. An easy and safe range for z would be between x and $k - 1$, plus $z = \infty$.

The answer is $S(y, 1, \infty)$.

2.10. Define

$S(k, i, j) =$ the maximum return from the remaining process if we start year k with a machine of age i, last overhauled at age j.

$$S(k, i, j) = \max \begin{bmatrix} -e(k, i, j) + n(k, 0, 0) + S(k + 1, 1, 0) \\ n(k, i, j) + S(k + 1, i + 1, j) \\ -o(k, i) + n(k, i, i) + S(k + 1, i + 1, i) \end{bmatrix}.$$

$$S(N + 1, i, j) = s(i, j).$$

2.11. Define S as in Problem 2.10.

$$S(k, i, j) = \max \begin{bmatrix} -e(k, i, j) + n(k, 0, 0) + S(k + 1, 1, 0) \\ n(k, i, j) + S(k + 1, i + 1, j) \\ -o(k, i) + S(k + 2, i + 2, i) \end{bmatrix}.$$

$$S(N + 1, i, j) = s(i, j).$$

Defining $S(N + 2, i, j) = -\infty$ assures that overhaul will not be commenced at the start of year N. This problem is similar to Problem 1.19.

2.12. *Backward procedure.* If hired as a consultant, we would need to know the current capital position in order to know what expenditures are permissible. Define

$S(k, i, j, C) =$ the maximum terminal capital plus salvage if we start year k with capital C and with a machine of age i, last overhauled at age j.

$$S(k, i, j, C) = \max \begin{bmatrix} S(k + 1, 1, 0, C - e(k, i, j) + n(k, 0, 0)) \\ S(k + 1, i + 1, j, C + n(k, i, j)) \\ S(k + 1, i + 1, i, C - o(k, i) + n(k, i, i)) \end{bmatrix},$$

where line 1 on the right (buy) is evaluated only if $C \geqslant e(k, i, j)$ and line 3 is evaluated only if $C \geqslant o(k, i)$. $S(N + 1, i, j, C) = C + s(i, j)$.

Forward procedure. Define

$T(k, i, j) =$ the maximum capital position attainable at the beginning of year k, starting year one with a new machine and starting year k with a machine of age i, last overhauled at age j.

If $i > 1$ and $j < i - 1$ (i.e., the machine was neither replaced nor overhauled the previous year), then

$$T(k, i, j) = n(k - 1, i - 1, j) + T(k - 1, i - 1, j).$$

If $i > 1$ and $j = i - 1$ (the machine was overhauled the previous year), then

$$T(k, i, i - 1) = -o(k - 1, i - 1) + n(k - 1, i - 1, i - 1) + \max_{l = 0, 1, \ldots, i - 2} [T(k - 1, i - 1, l)]$$

if the maximum T on the right equals or exceeds $o(k - 1, i - 1)$ (i.e., overhaul is affordable). Otherwise, $T(k, i, i - 1) = -\infty$ (i.e., the optimal path cannot go to this situation). Finally, if $i = 1$ and $j = 0$ (the machine was replaced the previous year), then

$$T(k, 1, 0) = n(k - 1, 0, 0) + \max_P [-e(k - 1, l, m) + T(k - 1, l, m)],$$

where P is the set of allowable ages and ages at last overhaul for which $T(k - 1, l, m) \geqslant e(k - 1, l, m)$ (i.e., purchase was affordable) and $T(k, 1, 0) = -\infty$ if P is empty.

The boundary condition is $T(1, 0, 0) = 0$. The answer is

$$\max_{i, j} [T(N + 1, i, j) + s(i, j)].$$

What is novel and very interesting about this problem is that the forward procedure requires fewer state variables than the backward procedure since the value of the optimal value function itself serves the purpose of one state variable. Were the constraint (quite unrealistically) that one could not purchase if the capital position *exceeded* some quantity, the state reduction in the forward procedure would not be correct since the overall optimal path might arrive at some intermediate situation with less than maximum possible capital in order to allow a machine replacement at that point.

2.13. Let

$S(i, j, k) =$ the minimum cost of getting from an i-year-old machine to a j-year-old one during a process lasting k years.

$$S(i, j, 2k) = \min_{l=1,\ldots,M} [S(i, l, k) + S(l, j, k)],$$

$$S(i, j, 1) = u(i, j-1) + c(j-1),$$

where $i = 1, \ldots, M$ and $j = 1, \ldots, M$.

Since i, j, and l can each take on M values, this procedure takes roughly NM^3 additions and NM^3 comparisons to solve a problem of duration 2^N stages.

The conventional dynamic-programming solution requires roughly $2^{N+1}M^2$ additions and $2^N M^2$ comparisons.

For any M and for sufficiently large N, doubling-up is more efficient, but it may not be for small problems. For $M = 10$ and $N = 4$ (i.e., 16 stages), the conventional procedure is preferable.

Chapter 3

3.1. $f_k(x) =$ the maximum return obtainable from activities 1 through k, given x units of resource to be allocated to activities 1 through k.

$$f_k(x) = \max_{x_k = 0, 1, \ldots, x} [r_k(x_k) + f_{k-1}(x - x_k)]. \tag{S3.1}$$

$$f_1(x) = r_1(x).$$

Let $p_k(x) =$ the x_k that maximizes the right-hand side of (S3.1) when $f_k(x)$ is computed.

$f_2(0) = 0, \qquad p_2(0) = 0;$

$f_2(1) = \max[0 + 3, 1 + 0] = 3, \qquad p_2(1) = 0;$

$f_2(2) = \max[0 + 7, 1 + 3, 2 + 0] = 7, \qquad p_2(2) = 0;$

$f_2(3) = 10, \qquad p_2(3) = 0;$

$f_2(4) = 12, \qquad p_2(4) = 0;$

$f_2(5) = 13, \qquad p_2(5) = 0, 1, \text{ or } 5;$

$f_2(6) = 17, \qquad p_2(6) = 6;$

$f_2(7) = 20, \qquad p_2(7) = 5 \text{ or } 6;$

$f_2(8) = 24, \qquad p_2(8) = 6.$

$f_3(0) = 0, \qquad p_3(0) = 0;$

$f_3(1) = \max[0 + 3, 2 + 0] = 3, \qquad p_3(1) = 0;$

$f_3(2) = \max[0 + 7, 2 + 3, 4 + 0] = 7, \qquad p_3(2) = 0;$

$f_3(3) = 10, \qquad p_3(3) = 0;$

$f_3(4) = 12, \qquad p_3(4) = 0 \text{ or } 1;$

$f_3(5) = 14, \qquad p_3(5) = 1 \text{ or } 2;$

$f_3(6) = 17, \qquad p_3(6) = 0;$

$f_3(7) = 20, \qquad p_3(7) = 0;$

$f_3(8) = 24, \qquad p_3(8) = 0.$

$f_4(8) = \max[0 + 24, 1 + 20, 3 + 17, 6 + 14, 9 + 12, 12 + 10, 14 + 7, 16 + 3, 17 + 0] = 24,$

$p_4(8) = 0.$

Optimal policy: $p_4(8) = 0$, $p_3(8) = 0$, $p_2(8) = 6$, $p_1(2) = 2$.

Each stage except the first and last requires $1 + 2 + \cdots + X + 1$ additions, so the same number of additions is required as for the backward procedure.

3.2. Definition (3.3) and recurrence relation (3.4) are unchanged. The boundary condition becomes

$$f_N(x) = \max_{x_N = 0, 1, \ldots, x} [r_N(x_N)]$$

or, equivalently,

$$f_{N+1}(x) = 0 \qquad \text{for} \quad x = 0, 1, \ldots, X,$$

since not all resources need be used.

3.3. Let $f_k(x) =$ the maximum return from an allocation of x units of resource to k activities.

$$f_{2k}(x) = \max_{y = 0, 1, \ldots, [x/2]} [f_k(y) + f_k(x - y)]. \tag{S3.2}$$

$[x/2]$ denotes the greatest integer $< x/2$, and y is the allocation to the first k of the $2k$ identical activities. The allocation y need only go to $[x/2]$ if, when we divide x into two parts for allocation to the first k and second k activities, we allocate the smaller part to the first k.

$$f_1(x) = r(x).$$

Let $p_k(x)$ be the y that maximizes the right-hand side of (S3.2). Then $p_1(x) = x$.

$$f_2(0) = 0, \qquad p_2(0) = 0;$$

$$f_2(1) = 2, \qquad p_2(1) = 0;$$

$$f_2(2) = \max[0 + 5, 2 + 2] = 5, \qquad p_2(2) = 0;$$

$$f_2(3) = \max[0 + 9, 2 + 5] = 9, \qquad p_2(3) = 0;$$

$$f_2(4) = \max[0 + 12, 2 + 9, 5 + 5] = 12, \qquad p_2(4) = 0;$$

$$f_2(5) = 14, \qquad p_2(5) = 0, 1, \text{ or } 2;$$

$$f_2(6) = 18, \qquad p_2(6) = 3;$$

$$f_2(7) = 21, \qquad p_2(7) = 3;$$

$$f_2(8) = 24, \qquad p_2(8) = 4;$$

$$f_2(9) = 26, \qquad p_2(9) = 4;$$

$$f_2(10) = 28, \qquad p_2(10) = 4 \text{ or } 5.$$

$$f_4(0) = 0, \qquad p_4(0) = 0;$$

$$f_4(1) = 2, \qquad p_4(1) = 0;$$

$$f_4(2) = \max[0 + 5, 2 + 2] = 5, \qquad p_4(2) = 0;$$

$$f_4(3) = \max[0 + 9, 2 + 5] = 9, \qquad p_4(3) = 0;$$

$$f_4(4) = 12, \qquad p_4(4) = 0;$$

$$f_4(5) = 14, \qquad p_4(5) = 0, 1, \text{ or } 2;$$

$$f_4(6) = 18, \qquad p_4(6) = 0 \text{ or } 3;$$

$$f_4(7) = 21, \qquad p_4(7) = 0 \text{ or } 3;$$

$$f_4(8) = 24, \qquad p_4(8) = 0 \text{ or } 4;$$

$$f_4(9) = 27, \qquad p_4(9) = 3;$$

$$f_4(10) = 30, \qquad p_4(10) = 3 \text{ or } 4.$$

$f_8(10) = \max[0 + 30, 2 + 27, 5 + 24, 9 + 21, 12 + 18, 14 + 14] = 30,$ $p_8(10) = 0, 3,$ or 4.

An optimal policy is obtained as follows: $p_8(10) = 0$ means nothing is allocated to four of the activities. $p_4(10) = 3$ means 3 is allocated to two of the remaining activities and 7 to the other two. $p_2(3) = 0$ means 0 to one activity and 3 to the other. $p_2(7) = 3$ means 3 to one activity and 4 to the other. The optimal solution is 4 to one activity, 3 to each of two different activities, and 0 to five activities. $p_8(10) = 3$ and 4 lead to other representations of the same policy.

Approximately $NX^2/4$ additions are needed for solution for 2^N activities. The denominator is 4 rather than 2 because of the reduced range of y in the maximization.

3.4. Let $f_k(x, y) = $ the maximum return obtainable from activities k through N, given x units of resource one and y units of resource two.

$$f_k(x, y) = \max_{\substack{x_k = 0, 1, \ldots, x \\ y_k = 0, 1, \ldots, y}} [r_k(x_k, y_k) + f_{k+1}(x - x_k, y - y_k)]. \tag{S3.3}$$

$$f_N(x, y) = r_N(x, y).$$

For given k, x, and y, $(x + 1)(y + 1)$ additions are needed. Hence, for given k,

$$\sum_{x=0}^{X} \sum_{y=0}^{Y} (x + 1)(y + 1) \approx \frac{X^2}{2} \frac{Y^2}{2}$$

additions are needed. Consequently, a total of roughly $NX^2Y^2/4$ additions are required for solution.

3.5. Let $p_k(x, y) = (x_k, y_k)$ where x_k, y_k maximizes the right-hand side of (S3.3).

	$f_2(x, y)$					$p_2(x, y)$ (not unique)			
x \ y	0	1	2	3	x \ y	0	1	2	3
0	0	3	5	8	0	(0, 0)	(0, 0)	(0, 0)	(0, 0)
1	2	5	7	9	1	(0, 0)	(0, 0)	(0, 0)	(0, 0)
2	4	7	9	12	2	(0, 0)	(0, 0)	(0, 0)	(2, 0)
3	6	9	11	14	3	(0, 0)	(0, 0)	(0, 0)	(3, 0)

$f_1(3, 3) = 16$, $p_1(3, 3) = (1, 0)$. Optimal policy is $x_1 = 1$, $y_1 = 0$; $x_2 = 2$, $y_2 = 0$; $x_3 = 0$, $y_3 = 3$.

3.6. Let $f_k(x, y, z) = $ the maximum return obtainable from activities k through N starting with x units of resource one, y units of resource two, and z units of resource three.

$$f_k(x, y, z) = \max_{\substack{x_k = 0, 1, \ldots, x \\ y_k = 0, 1, \ldots, y \\ z_k = 0, 1, \ldots, z}} [r_k(x_k, y_k, z_k) + f_{k+1}(x - x_k, y - y_k, z - z_k)].$$

$$f_N(x, y, z) = r_N(x, y, z).$$

The approximate number of additions is $NX^2Y^2Z^2/8$. This is a three-dimensional problem.

3.7. Renumber the activities so that the M specified activities are numbered 1 through M. Define $f_k(x)$ as in (3.3).

For $k = M + 1, M + 2, \ldots, N$, recurrence relation (3.4) applies and f must be computed only for $x = 0, 1, \ldots, X - Y$. The reason for this upper limit is that at least Y units must be allocated to the first M activities so at most $X - Y$ units can remain.

For $k = M$, if $x > X - Y$, x_M must be sufficiently large so that x at the start of stage

$M + 1$ is less than or equal to $X - Y$. Hence,

$$f_M(x) = \max_{x_M = x - (X - Y), \ldots, x} [r_M(x_M) + f_{M+1}(x - x_M)].$$

For $k = M$, $x \leq X - Y$ and $k < M$, $x \leq X$, recurrence relation (3.4) applies.

Solution requires roughly $MX^2/2 + (N - M)(X - Y)^2/2$ additions, which is fewer additions than in the solution of Section 1.

3.8. Let $f_j(x, k) =$ the maximum obtainable return from allocating x units to activities j through N where k ($< M$) of the activities 1 through $j - 1$ were used at nonzero level and at most M allocations can be nonzero.

$$f_j(x, k) = \max \left[\begin{array}{l} r_j(0) + f_{j+1}(x, k) \\ \max_{x_j = 1, \ldots, x} \left[r_j(x_j) + f_{j+1}(x - x_j, k + 1) \right] \end{array} \right],$$

where f equals $-\infty$ whenever its last argument exceeds M (i.e., where $k = M$, only a zero allocation is permissible).

$$f_N(x, k) = r_N(x) \quad \text{if} \quad x > 0 \text{ and } k < M,$$
$$f_N(0, k) = r_N(0) \quad \text{if} \quad k \leq M,$$
$$f_N(x, k) = -\infty \quad \text{if} \quad x > 0 \text{ and } k = M.$$

3.9. Let $f_j(x, k) =$ the maximum value of the criterion (including the h assessment) obtainable if you have x units to allocate to activities j through N and k of the allocations to activities 1 through $j - 1$ were nonzero.

$$f_j(x, k) = \max \left[\begin{array}{l} r_j(0) + f_{j+1}(x, k) \\ \max_{x_j = 1, 2, \ldots, x} \left[r_j(x_j) + f_{j+1}(x - x_j, k + 1) \right] \end{array} \right].$$
$$f_N(x, k) = r_N(x) - h(k + 1) \quad \text{if} \quad x > 0,$$
$$= r_N(0) - h(k) \quad \text{if} \quad x = 0.$$

Answer is $f_1(X, 0)$.

Since f_N must be computed for $k = 0, 1, \ldots, N - 1$ and, roughly, the usual range of x, f_{N-1} for $k = 0, 1, \ldots, N - 2$ and the usual x's, etc., solution takes approximately

$$(N^2/2)(X^2/2) = N^2 X^2/4$$

additions, which is $N/2$ (the average number of values of k) times the computation needed for the problem of Section 1.

3.10. Let $f_k(x, y) =$ the maximum dollar return obtainable from divisions k through N, using x units of resource and where y units of goodwill have been obtained from divisions 1 through $k - 1$ and where at least Y units must be obtained in all.

$$f_k(x, y) = \max_{x_k = 0, 1, \ldots, x} [r_k(x_k) + f_{k+1}(x - x_k, y + g_k(x_k))].$$
$$f_{N+1}(x, y) = 0 \quad \text{if} \quad y \geq Y,$$
$$f_{N+1}(x, y) = -\infty \quad \text{if} \quad y < Y.$$

The boundary conditions prohibit ending with total goodwill less than Y by assigning such a terminal state an unacceptable return.

The range of x is, as usual, $x = 0, 1, \ldots, X$. The range of y depends on the data. A conservative, and unnecessarily large, range is as follows. Let $\underline{g}_k$ be the maximum value of $g_k(x)$ and $\bar{\bar{g}}_k$ the minimum. Let $\bar{G}_k$ be $\sum_{i=1}^{k-1} \underline{g}_k$ and $\bar{\bar{G}}_k$ be $\sum_{i=1}^{k-1} \bar{\bar{g}}_k$. Then the range of y at

stage k equals $\overline{\overline{G}}_k$, $\overline{\overline{G}}_k + 1, \ldots, \overline{G}_k$. Let H_k denote the number of values of y at stage k. The approximate number of additions required is $\sum_{k=1}^{N} [X^2/2] H_k$.

3.11. Let

$$R_i(z) = \max_{x_i = 0, 1, \ldots, [z/3]} [r_i(x_i, z - 3x_i)]. \tag{S3.4}$$

Let $p_i(z) = (x_i, y_i)$ where x_i, y_i maximizes the right-hand side of (S3.4).

z	$R_1(z)$	$p_1(z)$	$R_2(z)$	$p_2(z)$	$R_3(z)$	$p_3(z)$	$R_4(z)$	$p_4(z)$
0	0	(0, 0)	0	(0, 0)	0	(0, 0)	0	(0, 0)
1	5	(0, 1)	3	(0, 1)	1	(0, 1)	1	(0, 1)
2	9	(0, 2)	5	(0, 2)	3	(0, 2)	3	(0, 2)
3	11	(0, 3)	6	(0, 3)	6	(0, 3)	7	(1, 0)
4	12	(0, 4)	7	(0, 4)	10	(0, 4)	10	(0, 4), (1, 1)
5	13	(0, 5), (1, 2)	9	(1, 2)	12	(0, 5)	14	(1, 2)
6	18	(1, 3)	12	(1, 3)	13	(0, 6)	17	(1, 3)

$$f_k(z) = \max_{z_k = 0, 1, \ldots, z} [R_k(z_k) + f_{k+1}(z - z_k)]. \tag{S3.5}$$

(We shall sometimes omit the definition of the optimal value function when its meaning is fairly obvious. The student should *never* do this!)

$f_4(z) = R_4(z)$. $q_k(z) = z_k$ which maximizes the right-hand side of (S3.5).

z	$f_3(z)$	$q_3(z)$	$f_2(z)$	$q_2(z)$	$f_1(z)$	$q_1(z)$
0	0	0	0	0		
1	1	0, 1	3	1		
2	3	0, 2	5	2		
3	7	0	7	0		
4	10	0, 4	10	0, 1		
5	14	0	14	0		
6	17	0	17	0, 1	19	1, 2

There are five solutions. One is $q_1(6) = 1$ ($x_1 = 0$, $y_1 = 1$), $q_2(5) = 0$ ($x_2 = 0$, $y_2 = 0$), $q_3(5) = 0$ ($x_3 = 0$, $y_3 = 0$), $q_4(5) = 5$ ($x_4 = 1$, $y_4 = 2$).

3.12.

$$R_i(w) = \max_{\substack{x_i = 0, 1, \ldots, [w/a] \\ y_i = 0, 1, \ldots, [(w - ax_i)/b]}} \left[r_i \left(x_i, y_i, \left[\frac{w - ax_i - by_i}{c} \right] \right) \right].$$

$$f_k(w) = \max_{w_k = 0, 1, \ldots, w} [R_k(w_k) + f_{k+1}(w - w_k)].$$

For computing the R's when $a = b = c = 1$, roughly $NW^3/6$ comparisons are needed. For computing the f's, roughly $NW^2/2$ additions and a like number of comparisons are needed.

3.13.

$$r'_k(x_k) = \max_{y_k = 0, 1, 2, 3} [r_k(x_k, y_k) - \tfrac{3}{2} y_k].$$

Let $p_k(x) = $ the y_k that maximizes the right-hand side of $r'_k(x)$.

x	$r'_1(x)$	$p_1(x)$	$r'_2(x)$	$p_2(x)$	$r'_3(x)$	$p_3(x)$
0	1.5	3	1.5	3	3.5	3
1	4	0	3	2	4.5	3
2	5	0	5	2	6.5	3
3	6	0	7	2	8.5	3

$$f_k(x) = \max_{x_k = 0, 1, 2, 3} [r'_k(x_k) + f_{k+1}(x - x_k)]. \tag{S3.6}$$

Let $q_k(x) = $ the x_k that maximizes the right-hand side of (S3.6).

x	$f_3(x)$	$q_3(x)$	$f_2(x)$	$q_2(x)$	$f_1(x)$	$q_1(x)$
0	3.5	0	5	0		
1	4.5	1	6.5	1		
2	6.5	2	8.5	2		
3	8.5	3	10.5	3	12.5	1

The optimal solution is $x_1 = 1$, $y_1 = 0$; $x_2 = 2$, $y_2 = 2$; $x_3 = 0$, $y_3 = 3$. This solution yields a return of $4 + 8 + 8 = 20$. The sum of the y's is 5. Hence, return $-(\frac{3}{2} \cdot$ sum of y's$) = 12.5 = f_1(3)$. The problem has been solved for $Y = 5$.

3.14.

Path	Cost, v	Turns, u
$UUDD$	8	1
$UDUD$	5	3
$UDDU$	8	2
$DUUD$	6	2
$DUDU$	9	3
$DDUU$	11	1

λ in the range $-2 < \lambda < -1$ will solve the two-turn problem. Let $\lambda = -1.5$. Solve for the minimum-cost path where each turn costs 1.5.

$S(x, y, z) = $ minimum-cost path from (x, y) to $(4, 0)$ where $z = 0$ means you arrived at (x, y) going up and $z = 1$ means you arrived going down.

$$S(x, y, 0) = \min \begin{bmatrix} a_u(x, y) + S(x + 1, y + 1, 0) \\ a_d(x, y) + 1.5 + S(x + 1, y - 1, 1) \end{bmatrix},$$

$$S(x, y, 1) = \min \begin{bmatrix} a_u(x, y) + 1.5 + S(x + 1, y + 1, 0) \\ a_d(x, y) + S(x + 1, y - 1, 1) \end{bmatrix}.$$

$$S(4, 0, 0) = S(4, 0, 1) = 0.$$

Let $p(x, y, z) = $ optimal direction to go.

$$\begin{aligned} S(3, 1, 0) &= 2.5, & p(3, 1, 0) &= D; \\ S(3, 1, 1) &= 1, & p(3, 1, 1) &= D; \\ S(3, -1, 0) &= 3, & p(3, -1, 0) &= U; \\ S(3, -1, 1) &= 4.5, & p(3, -1, 1) &= U. \end{aligned}$$
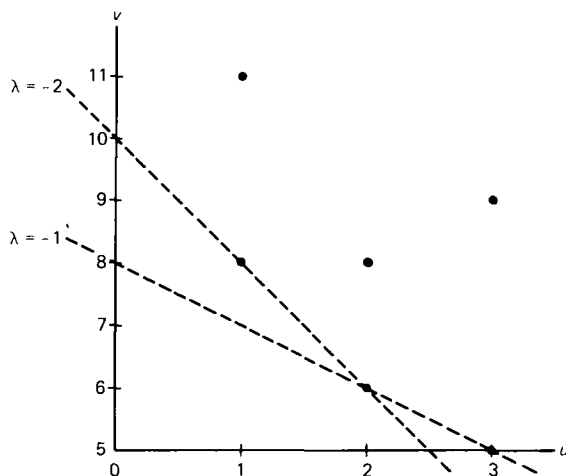
Figure S3.1

$$S(2, 2, 0) = 5.5, \qquad p(2, 2, 0) = D;$$
$$S(2, 0, 0) = 3.5, \qquad p(2, 0, 0) = U;$$
$$S(2, 0, 1) = 5, \qquad p(2, 0, 1) = U;$$
$$S(2, -2, 1) = 6.5, \qquad p(2, -2, 1) = U.$$

$$S(1, 1, 0) = 8.5, \qquad p(1, 1, 0) = U \text{ or } D;$$
$$S(1, -1, 1) = 7, \qquad p(1, -1, 0) = U.$$

The answer is the minimum of $9\frac{1}{2}$ and 9 which equals 9, and the decision is D. The optimal path is D–U–U–D. The path has exactly two turns, as expected, and costs 9, including the turn costs, or six if the turn costs are refunded.

3.15. See Figure S3.2.

In the case $y_1 + y_2 = 3$, the λ that minimizes $f(\lambda)$, $\lambda = 2$, is the Lagrange multiplier that solves the numerical example for $Y = 3$. In the case of $y_1 + y_2 = 1$, $\lambda = 2.5$ minimizes $f(\lambda)$ and $\sum_{i=1}^{2} y_i (2.5) = 0$ or 2, rather than 1. This means that $y_1 + y_2 = 1$ is in a gap (see step 4d of the Lagrange multiplier procedure).

It can be shown that $f(\lambda)$ is a convex (and thus continous) function of λ. Furthermore, if we minimize $f(\lambda)$ over λ and $\sum_{i=1}^{N} y_i(\lambda) = Y$, then we have the solution to the original problem. Otherwise, a gap exists. Therefore, the search for the correct value of λ can be conducted by searching for the value of λ that minimizes $f(\lambda)$. Note also that $f(\lambda) = f_1(X; \lambda) + \lambda Y$, where $f_1(X; \lambda)$ is the optimal value function which is computed from (3.11).

3.16. If a line with positive slope last touches a point of the plot (see, for example, Figure 3.2) at the highest point on the line $u = Y$, then v must be less than its maximum value at $u = Y$ for all $u < Y$.

3.17. Let $f_k(x, y) = $ the maximum value of the criterion from stage k until the end where $x_k = y$ and where $\sum_{i=k+1}^{N} x_i = x$.
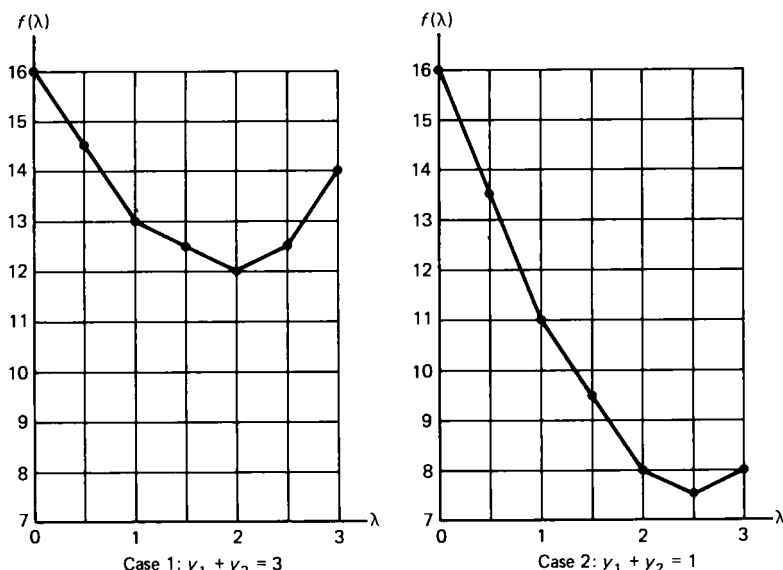
Figure S3.2

$$f_k(x, y) = \max_{x_{k+1} = 0, 1, \ldots, x} [g_k(y, x_{k+1}) + f_{k+1}(x - x_{k+1}, x_{k+1})].$$

$$f_{N-1}(x, y) = g_{N-1}(y, x).$$

The answer is $\max_{y = 0, 1, \ldots, x} f_0(X - y, y)$.

To use a Lagrange multiplier, one might neglect the constraint, $\sum_{i=1}^{N} x_i = X$, and subtract $\lambda \sum_{i=0}^{N} x_i$ from the criterion. Define $f_k(y; \lambda) =$ the maximum value of the modified criterion, given you start stage k with $x_k = y$.

$$f_k(y; \lambda) = \max_{x_{k+1} = 0, 1, \ldots, X} [g_k(y, x_{k+1}) - \lambda x_{k+1} + f_{k+1}(x_{k+1}; \lambda)].$$

$$f_{N-1}(y; \lambda) = \max_{x_N = 0, 1, \ldots, X} [g_{N-1}(y, x_N) - \lambda x_N].$$

The answer is the maximum over y of $[f_0(y; \lambda) - \lambda y]$. A value of λ is sought such that the resulting x_i sum to X.

3.18. If neglecting the goodwill constraint and solving results in too little goodwill, try to find a λ such that a modified criterion giving a bonus of λ per unit of goodwill results in total goodwill Y. Let $f_k(x; \lambda) =$ the maximum value of the modified criterion starting with $\$x$ and using divisions k through N.

$$f_k(x; \lambda) = \max_{x_k = 0, 1, \ldots, x} [r_k(x_k) + \lambda g_k(x_k) + f_{k+1}(x - x_k; \lambda)].$$

$$f_{N+1}(x; \lambda) = 0.$$

A positive bonus λ is sought so that the solution gives total goodwill of Y. Note that changing the sense of the inequality in the constraint changes the sign of the multiplier from that in the problem of Section 9 of the chapter.

For fixed λ, about NX^2 additions are required.

3.19. Let the modified criterion be the original one minus $\lambda\sum_{i=1}^{2^N} y_i$ and neglect the y constraint. Define $r'(x) = \max_{y=0, 1, \ldots, x}[r(x, y) - \lambda y]$. Define $f_k(x; \lambda) =$ the maximum value of the modified criterion using k activities and allocating x units of resource one.

$$f_{2k}(x; \lambda) = \max_{z=0, 1, \ldots, \lfloor x/2\rfloor}[f_k(z; \lambda) + f_k(x - z; \lambda)].$$

$$f_1(x; \lambda) = r'(x).$$

A value of λ is sought such that when the optimal x's are found, the associated y's sum to Y.

Each value of λ requires roughly $NX^2/4$ additions to compute f and XY additions to compute r'. For $N = 5$ (32 activities), $X = 1000$ and $Y = 1000$, roughly $\frac{5}{4} \cdot 10^6 + 10^6$ additions are needed.

Chapter 4

4.1. Assume the contrary; that is, assume every node has at least one inward-pointing arc. Arbitrarily choose a node. Call it node i_1. Node i_1 must have an inward-pointing arc. This arc must emanate from some node, say node i_2. Node i_2 must have an inward-pointing arc. If this arc emanated from node i_1, then a cycle would exist. Therefore, this arc must emanate from some other node, say node i_3. Continuing in this manner, node i_{N-1} must have an inward-pointing arc. This arc must emanate from node i_N since, otherwise, a cycle would exist. Finally, node i_N must have an inward-pointing arc. Since all nodes have been used, this arc must emanate from one of the nodes $i_1, i_2, \ldots, i_{N-1}$. However, this implies that a cycle exists, which is contrary to our assumption that the network is acyclic. Therefore, there must be at least one node with only outward-pointing arcs.

4.2. See Figure S4.1.

4.3.

$f_1 = 0$;

$f_2 = f_1 + d_{12} = 0 + 4 = 4, \qquad p_2 = 1$;

$f_3 = \min[f_1 + d_{13}, f_2 + d_{23}] = \min[0 + 3, 4 + 2] = 3, \qquad p_3 = 1$;

$f_4 = \min[f_1 + d_{14}, f_2 + d_{24}, f_3 + d_{34}] = \min[0 + 5, 4 + \infty, 3 + 1] = 4, \qquad p_4 = 3$;

$f_5 = \min[f_3 + d_{35}, f_4 + d_{45}] = \min[3 + 5, 4 + 2] = 6, \qquad p_5 = 4$;

$f_6 = \min[f_2 + d_{26}, f_3 + d_{36}, f_5 + d_{56}] = \min[4 + 6, 3 + 7, 6 + 3] = 9, \qquad p_6 = 5$.
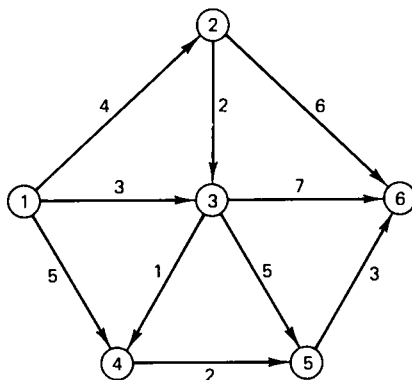

Figure S4.1

4.4. Enumerate the nodes such that if $(i, j) \in A$, then $i < j$. Define $f_i =$ the length of the shortest path from node i to node N.

$$f_i = \min_{j>i} [d_{ij} + f_j] \qquad (i = N - 1, N - 2, \ldots, 1),$$
$$f_N = 0.$$

4.5. We shall give an intuitive proof. (A rigorous proof requires using mathematical induction on both i and k.) Suppose that we seek the length of the shortest path from node 1 to node k among all paths having $i - 1$ or fewer decreases. Let node j be the node which immediately precedes node k. If $j < k$, then a decrease does not occur in going from node j to node k. Therefore, $i - 1$ or fewer decreases are allowed from node 1 to node j and $g_i(j) + d_{jk}$ is the length of the shortest such path. If $j > k$, then a decrease occurs in going from node j to node k. Therefore, $i - 2$ or fewer decreases are allowed from node 1 to node j and $g_{i-1}(j) + d_{jk}$ is the length of the shortest such path. It is clear that $g_i(k)$ is correctly given by (4.14).

4.6.

$$
\begin{array}{llll}
g_0(1) = 0, & g_1(1) = 0, & g_2(1) = 0, & g_3(1) = 0, \\
g_0(2) = \infty, & g_1(2) = 2, & g_2(2) = 1, & g_3(2) = 1, \\
g_0(3) = \infty, & g_1(3) = 5, & g_2(3) = 5, & g_3(3) = 5, \\
g_0(4) = \infty. & g_1(4) = 5. & g_2(4) = 4. & g_3(4) = 4.
\end{array}
$$

4.7. Let $F_i(k) = \{$all paths from node 1 to node k with i or fewer arcs$\}$ and let $G_i(k) = \{$all paths from node 1 to node k with $i - 1$ or fewer decreases$\}$. Suppose that $p \in F_i(k)$. Since a path must have at least two arcs to have a decrease, it follows that p has $i - 1$ or fewer decreases and thus $p \in G_i(k)$. Therefore, $F_i(k) \subseteq G_i(k)$ and every path considered in computing $f_i(k)$ is also considered in computing $g_i(k)$. It follows that $g_i(k) \leqslant f_i(k)$.

4.8. We shall give an intuitive argument. Clearly, the computation of $h_1(k)$ will examine all increasing sequences of nodes leading to each node k. Since only $j < k$ is considered, no decrease is allowed. When $h_2(k)$ is computed, all decreasing sequences (following the original increasing sequence) are considered but $j > k$ prohibits any increase after a decrease, so we have the best path with one reversal. When $h_3(k)$ is computed, increasing sequences are appended to the previous best paths, so two reversals are allowed. $h_4(k)$ then allows a further decreasing sequence, hence three reversals, etc.

4.9. We shall first show that if $h_i(k) = h_{i-1}(k)$ $(k = 1, 2, \ldots, N)$ for some i, $2 \leqslant i \leqslant N$, then $h_{i+1}(k) = h_i(k)$ $(k = 1, 2, \ldots, N)$. This implies that $h_i(k)$ is the length of the shortest path from node 1 to node k. The proof is by mathematical induction on k. Assume that $i + 1$ is odd. (The proof for $i + 1$ even is similar.) Consider first the case $k = 1$. It is easy to see that

$$h_{i+1}(1) = \min \left[\begin{array}{c} h_i(1) \\ \min_{j<1} \{ h_{i+1}(j) + d_{j1} \} \end{array} \right] = h_i(1).$$

Assume now that $h_{i+1}(l) = h_i(l)$ for $l = 1, 2, \ldots, k - 1$. We must show that $h_{i+1}(k) = h_i(k)$. From our inductive hypothesis, $h_i(k) = h_{i-1}(k)$, and the fact that $h_i(k)$ is a nonincreasing function of i, we get

$$h_{i+1}(k) = \min \left[\begin{array}{c} h_i(k) \\ \min_{j<k} \{ h_{i+1}(j) + d_{jk} \} \end{array} \right] = \min \left[\begin{array}{c} h_i(k) \\ \min_{j<k} \{ h_{i-1}(j) + d_{jk} \} \end{array} \right]$$
$$\geqslant \min \left[\begin{array}{c} h_i(k) \\ h_{i-1}(k) \end{array} \right] = h_i(k) \geqslant h_{i+1}(k).$$

Therefore, $h_{i+1}(k) = h_i(k)$.

No shortest path can have more than $N - 2$ reversals ($N - 1$ iterations). Therefore, if $h_N(k) \neq h_{N-1}(k)$ for some k, then a negative cycle exists.

4.10.

$$h_0(1) = 0, \quad h_1(1) = 0, \quad h_2(1) = 0, \quad h_3(1) = 0, \quad h_4(1) = 0,$$
$$h_0(2) = \infty, \quad h_1(2) = 2, \quad h_2(2) = 1, \quad h_3(2) = 1, \quad h_4(2) = 1,$$
$$h_0(3) = \infty, \quad h_1(3) = 5, \quad h_2(3) = 5, \quad h_3(3) = 5, \quad h_4(3) = 5,$$
$$h_0(4) = \infty. \quad h_1(4) = 5. \quad h_2(4) = 5. \quad h_3(4) = 4. \quad h_4(4) = 4.$$

4.11. Let $G_i(k) = \{$all paths from node 1 to node k with $i - 1$ or fewer decreases$\}$ and let $H_{2i-1}(k) = \{$all paths from node 1 to node k with $2(i - 1)$ or fewer reversals$\}$. Suppose $p \in G_i(k)$. Since no decrease can be part of more than two reversals (if $i_p < i_{p-1}$, then reversals may occur at i_{p-1} and i_p), it follows that p has $2(i - 1)$ or fewer reversals and thus $p \in H_{2i-1}(k)$. Therefore, $G_i(k) \subseteq H_{2i-1}(k)$. It follows that $h_{2i-1}(k) \leqslant g_i(k)$.

If procedure 2 converges in i iterations, then $g_i(k) = g_{i-1}(k)$. This in turn implies that $h_{2i-1}(k) = h_{2i-2}(k) = h_{2i-3}(k)$. It follows from Problem 4.9 that procedure 3 must converge in no more than $2i - 2$ iterations.

4.12. We give only the answers.

$$f_6(j, k)$$

j \ k	1	2	3	4	5	6
1	0	1	1	-1	6	4
2	1	0	0	-2	5	3
3	2	3	0	1	6	6
4	4	3	2	0	7	5
5	-3	-2	-5	-4	0	1
6	0	1	-2	-1	3	0

4.13. The computation of $g_i(j, i)$ requires $(i - 1)^2$ additions and $(i - 1)(i - 2)$ comparisons. Similarly for $g_i(i, k)$. The computation of $g_i(i, i)$ requires $i - 1$ additions and $i - 1$ comparisons. Finally, the computation of $g_i(j, k)$ requires $(i - 1)(i - 2)$ additions and $(i - 1)(i - 2)$ comparisons. If we sum these numbers from $i = 2$ to $i = N$, we get the stated number of operations.

4.14. The answers are the same as for Problem 4.12.

4.15. Use a Dijkstra-like procedure. Let $\overline{\Pi}(i)$, $\Pi(i)$, and $N(1)$ be defined similarly to the way they were defined in Section 3A. Initially $N(1) = \{1\}$. All other nodes have temporary labels, where $\Pi(i) = d_{1i}$. The algorithm is as follows:

Step 1: Determine a node with a temporary label, say node k, such that $\Pi(k) = \min_{i \notin N(1)}\Pi(i)$. Set $\overline{\Pi}(k) = \Pi(k)$ and $N(1) := N(1) \cup \{k\}$.

Step 2: If $N(1) = N$, then stop. If not, go to step 3.

Step 3: For each arc (k, l) (k is the node determined in step 1) such that $l \notin N(1)$, $\Pi(l) := \min[\Pi(l), \max\{\overline{\Pi}(k), d_{kl}\}]$. Go to step 1.

4.16. Use a Dijkstra-like procedure. Let $\overline{\Pi}(i)$, $\Pi(i)$, and $N(1)$ be defined similarly to the way they were defined in Section 3A. Initially $N(1) = \{1\}$ and $\overline{\Pi}(1) = 1$. All other nodes have temporary labels, where $\Pi(i) = p_{1i}$. The algorithm is as follows:

Step 1: Determine a node with a temporary label, say node k, such that $\Pi(k) = \max_{i \notin N(1)}\Pi(i)$. Set $\overline{\Pi}(k) = \Pi(k)$ and $N(1) := N(1) \cup \{k\}$.

Step 2:   If $N(1) = N$, then stop. If not, go to step 3.

Step 3:   For each arc (k, l) (k is the node determined in step 1) such that $l \notin N(1)$, $\Pi(l)$ $:= \max[\Pi(l), \overline{\Pi}(k) \cdot p_{kl}]$. Go to step 1.

4.17.   Use Dijkstra's procedure with step 3 modified as follows:

Step 3:   For each arc (k, l) (k is the node determined in step 1) such that $l \notin N(1)$, $\Pi(l)$ $:= \min[\Pi(l), \overline{\Pi}(k) + d_{kl}(\overline{\Pi}(k))]$. Go to step 1.

When the procedure terminates, $\overline{\Pi}(i)$ is the earliest possible arrival time at node i when you leave node 1 at time 0.

4.18.   The procedure is almost the same as in the normal case. However, the number of labels associated with a node is equal to the number of arcs leading into the node. The procedure terminates when any one of the labels associated with node 9 becomes permanent.

The length of the shortest path from node 1 to 9 is 15 and the associated shortest path is 1–3–5–7–9.

4.19.   Create two dummy nodes, say node 0 and node $N + 1$. Let

$$d_{0i} = \begin{bmatrix} 0 & \text{if} & i \in A, \\ \infty & \text{if} & i \notin A; \end{bmatrix} \quad \text{and} \quad d_{i, N+1} = \begin{bmatrix} 0 & \text{if} & i \in B, \\ \infty & \text{if} & i \notin B. \end{bmatrix}$$

All other arcs are infinite (i.e., d_{i0} and $d_{N+1, i}$).

Use Dijkstra's procedure to find the length of the shortest path from node 0 to node $N + 1$. The number of operations required is the same as for the normal problem with $N + 2$ nodes.

Chapter 5

5.1.   Consider the sequence of cities which makes up the optimal path. It begins and ends at city 1 and, furthermore, each city appears in the sequence at least once. Circle the city 1 at the beginning of the sequence, circle the city 1 at the end of the sequence, and also circle each other city exactly once. The sum of the d_{ij} between any two circled cities must equal the length of the shortest path between the two circled cities. (Otherwise, this sequence of cities could not be the optimal path.) When viewed in this manner, it is clear that the optimal sequence of cities is one of the sequences considered by the procedure in question.

5.2.   Define $f_i(j, S) =$ the length of the shortest path from city 1 to city j via the set of i intermediate cities S.

$$f_i(j, S) = \min_{k \in S} [f_{i-1}(k, S - \{k\}) + d_{k, j, i-1}] \qquad (i = 1, 2, \dots, N - 2).$$

$$f_0(j, -) = d_{1, j, 0}.$$

The answer is $\min_{j=2, 3, \dots, N} [f_{N-2}(j, S) + d_{j, 1, N-2}]$. This problem requires more data than the problem of Section 2; however, the two problems have exactly the same computational requirements. As a matter of fact, the amount of computation is the same even if d_{ij} depends on the actual set of cities visited before city i.

5.3.   Define $f_i(j; S, k) =$ the length of the shortest path from city 1 to city j via the set of $i - 1$ intermediate cities S and also the intermediate city k (the last intermediate city).

$$f_i(j; S, k) = \min_{l \in S} [f_{i-1}(k; S - \{l\}, l) + d_{k, j, l}] \qquad (i = 2, 3, \dots, N - 2).$$

$$f_1(j; -, k) = d_{1k} + d_{k, j, 1}.$$

The answer is

$$\min_{\substack{j=2, 3, \ldots, N \\ k=2, 3, \ldots, N \\ j \neq k}} [f_{N-2}(j; S, k) + d_{j, 1, k}].$$

5.4. Create a dummy city, say city 0. Let $d_{0i} = 0$ for $i = 1, 2, \ldots, N$. Solve the $(N + 1)$-city traveling-salesman problem where you start at city 0 and then visit each of the other N cities once and only once.

5.5. Let the N sets be denoted by $S_1, S_2, \ldots, S_N$. Let T be a set of i of these N sets. Define $f_i(k_j, T) = $ the length of the shortest path from city A to city j in set k which visits one and only one city in each of the i sets in T along the way.

$$f_i(k_j, T) = \min_{S_i \in T} \left\{ \min_{m=1, 2, \ldots, M} \left[f_{i-1}(l_m, T - \{S_i\}) + d_{l_m, k_j} \right] \right\}$$

$(i = 1, 2, \ldots, N - 1; \quad k = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M;$ all sets of i sets of cities T).

$$f_0(k_j, \_) = d_{A, k_j}.$$

The answer is $\min_{k=1, 2, \ldots, N}[\min_{j=1, 2, \ldots, M}\{f_{N-1}(k_j, T) + d_{k_j, B}\}]$.

5.6. Let N_j and S be as defined in Section 2, and let P_j be the set of all cities (other than city 1) that must precede city j. Define $f_i(j, S) = $ the length of the shortest path from city 1 to city j via the set of i intermediate cities S, and satisfying the given precedence relations.

$$f_i(j, S) = \begin{cases} \min_{k \in S} [f_{i-1}(k, S - \{k\}) + d_{kj}] & \text{if} \quad P_j \subseteq S, \\ \infty & \text{if} \quad P_j \nsubseteq S. \end{cases}$$

$$f_0(j, S) = \begin{cases} d_{1j} & \text{if} \quad P_j = \emptyset, \\ \infty & \text{otherwise}. \end{cases}$$

The answer is $\min_{j=2, 3, \ldots, N}[f_{N-2}(j, S) + d_{j1}]$. The amount of computation is decreased since for some sets, S, no calculations are necessary. The numerical solution for the problem in Figure 5.1 is as follows:

$f_0(2, \_) = \infty,$ $f_1(2, \{3\}) = 2 + 3 = 5(3),$ $f_2(2, \{3, 4\}) = \min[2 + 3, 3 + 1] = 4(4),$

$f_0(3, \_) = 2(1),$ $f_1(2, \{4\}) = \infty,$ $f_2(3, \{2, 4\}) = \min[\infty, \infty] = \infty,$

$f_0(4, \_) = 1(1).$ $f_1(3, \{2\}) = \infty,$ $f_2(4, \{2, 3\}) = \min[5 + 1, \infty] = 6(2).$

                 $f_1(3, \{4\}) = 1 + 1 = 2(4),$

                 $f_1(4, \{2\}) = \infty,$

                 $f_1(4, \{3\}) = 2 + 1 = 3(3).$

Answer = $\min[4 + 1, \infty, 6 + 1] = 5(2)$. The optimal path is 1–3–4–2–1.

5.7. Let $\overline{\Pi}(i, S)$ be the permanent (true) shortest distance from city 1 to city i via the set of intermediate cities S. ($\Pi(i, S)$ is the comparable temporary shortest distance.) Furthermore, let P be the set of pairs (i, S) with permanent shortest distances. Initially $\Pi(i, \_) = d_{1i}$. All other (i, S) pairs have temporary shortest distances equal to ∞. Then the Dijkstra-like procedure for this version of the traveling-salesman problem is as follows:

    Step 1: Determine a pair with a temporary label, say the pair (k, T), such that $\Pi(k, T) = \min_{(i, S) \notin P} \Pi(i, S)$. Set $\overline{\Pi}(k, T) = \Pi(k, T)$ and $P := P \cup \{(k, T)\}$.

    Step 2: If $T \cup \{1, k\} = N$, then stop. If not, go to step 3.

    Step 3: For each arc $(k, l) \in A$ (k is the city determined in step 1) such that

$l \notin T \cup \{1, k\}$, $\Pi(l, T \cup \{k\}) := \min[\Pi(l, T \cup \{k\}), \overline{\Pi}(k, T) + d_{kl}]$. Go to step 1.

The numerical solution is as follows:.

$\Pi(2, \_) = 1$, $\Pi(3, \_) = 2$, $\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$.

$\overline{\Pi}(2, \_) = 1$;
$\Pi(3, \_) = 2$, $\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(3, \{2\}) = 2$, $\Pi(4, \{2\}) = 2$, $\Pi(5, \{2\}) = 11$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$;
$\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(3, \{2\}) = 2$, $\Pi(4, \{2\}) = 2$, $\Pi(5, \{2\}) = 11$;
$\Pi(2, \{3\}) = 3$, $\Pi(4, \{3\}) = 12$, $\Pi(5, \{3\}) = 12$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$, $\overline{\Pi}(3, \{2\}) = 2$;
$\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(4, \{2\}) = 2$, $\Pi(5, \{2\}) = 11$;
$\Pi(2, \{3\}) = 3$, $\Pi(4, \{3\}) = 12$, $\Pi(5, \{3\}) = 12$;
$\Pi(4, \{2, 3\}) = 12$, $\Pi(5, \{2, 3\}) = 12$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$, $\overline{\Pi}(3, \{2\}) = 2$, $\overline{\Pi}(4, \{2\}) = 2$;
$\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(5, \{2\}) = 11$;
$\Pi(2, \{3\}) = 3$, $\Pi(4, \{3\}) = 12$, $\Pi(5, \{3\}) = 12$;
$\Pi(4, \{2, 3\}) = 12$, $\Pi(5, \{2, 3\}) = 12$;
$\Pi(3, \{2, 4\}) = 12$, $\Pi(5, \{2, 4\}) = 3$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$, $\overline{\Pi}(3, \{2\}) = 2$, $\overline{\Pi}(4, \{2\}) = 2$, $\overline{\Pi}(2, \{3\}) = 3$;
$\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(5, \{2\}) = 11$;
$\Pi(4, \{3\}) = 12$, $\Pi(5, \{3\}) = 12$;
$\Pi(4, \{2, 3\}) = \min[12, 4] = 4$, $\Pi(5, \{2, 3\}) = \min[12, 13] = 12$;
$\Pi(3, \{2, 4\}) = 12$, $\Pi(5, \{2, 4\}) = 3$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$, $\overline{\Pi}(3, \{2\}) = 2$, $\overline{\Pi}(4, \{2\}) = 2$, $\overline{\Pi}(2, \{3\}) = 3$,
$\overline{\Pi}(5, \{2, 4\}) = 3$;
$\Pi(4, \_) = 10$, $\Pi(5, \_) = 10$;
$\Pi(5, \{2\}) = 11$;
$\Pi(4, \{3\}) = 12$, $\Pi(5, \{3\}) = 12$;
$\Pi(4, \{2, 3\}) = 4$, $\Pi(5, \{2, 3\}) = 12$;
$\Pi(3, \{2, 4\}) = 12$;
$\Pi(3, \{2, 4, 5\}) = 13$.

$\overline{\Pi}(2, \_) = 1$, $\overline{\Pi}(3, \_) = 2$, $\overline{\Pi}(3, \{2\}) = 2$, $\overline{\Pi}(4, \{2\}) = 2$, $\overline{\Pi}(2, \{3\}) = 3$,
$\overline{\Pi}(5, \{2, 4\}) = 3$, $\overline{\Pi}(4, \{2, 3\}) = 4$;

$$\Pi(4, \_) = 10, \quad \Pi(5, \_) = 10;$$
$$\Pi(5, \{2\}) = 11;$$
$$\Pi(4, \{3\}) = 12, \quad \Pi(5, \{3\}) = 12;$$
$$\Pi(5, \{2, 3\}) = 12;$$
$$\Pi(3, \{2, 4\}) = 12;$$
$$\Pi(3, \{2, 4, 5\}) = 13;$$
$$\Pi(5, \{2, 3, 4\}) = 5.$$

Finally, $\overline{\Pi}(5, \{2, 3, 4\}) = 5$ and, thus, the answer to the problem is 5.

Chapter 6

6.1. $J = 36$.

6.2. $y(0) = -3$, $y(1) = -1$, $y(2) = -1$. Hence $x(1) = .5$, $x(2) = 1$, $x(3) = 2$. $J = 18$.

6.3. $p(3) = \frac{1}{4}$, $\quad p(2) = 4$, $\quad p(1) = 36$, $\quad p(0) = \frac{9}{2}$.

$$y(0) = -\frac{(36)\left(\frac{1}{2}\right)\left(\frac{1}{6}\right)(2)}{1 + 36\left(\frac{1}{36}\right)} = -3.$$

$$x(1) = \frac{1}{2}. \qquad y(1) = -\frac{(4)(3)\left(\frac{1}{2}\right)\left(\frac{1}{2}\right)}{2 + (4)\left(\frac{1}{4}\right)} = -1.$$

$$x(2) = 1, \quad y(2) = -1, \qquad x(3) = 2. \qquad J = 18.$$

$$V_0(2) = p(0)x^2(0) = \left(\frac{9}{2}\right)(4) = 18.$$

6.4.

$$V_i(x) = \min_{y}\{a(i)x^2 + b(i)xy + c(i)y^2 + d(i)x + e(i)y + f + p(i+1)[g(i)x+h(i)y + k(i)]^2$$

$$+ q(i+1)[g(i)x + h(i)y + k(i)] + r(i+1)\}.$$

$$y = -\frac{[b(i) + 2p(i+1)g(i)h(i)]x + e(i) + 2p(i+1)h(i)k(i) + q(i+1)h(i)}{2[c(i) + p(i+1)h^2(i)]}.$$

Hence

$$V_i(x) = \left\{ a(i) + p(i+1)g^2(i) - \frac{[b(i) + 2p(i+1)g(i)h(i)]^2}{4[c(i) + p(i+1)h^2(i)]} \right\} x^2$$

$$+ \left\{ d(i) + 2p(i+1)k(i)g(i) + q(i+1)g(i) \right.$$

$$\left. - \frac{[b(i) + 2p(i+1)g(i)h(i)][e(i) + 2p(i+1)h(i)k(i) + q(i+1)h(i)]}{2[c(i) + p(i+1)h^2(i)]} \right\} x$$

$$+ \left\{ f(i) + p(i+1)k^2(i) + q(i+1)k(i) + r(i+1) \right.$$

$$\left. - \frac{[e(i) + 2p(i+1)h(i)k(i) + q(i+1)h(i)]^2}{4[c(i) + p(i+1)h^2(i)]} \right\}$$

$$= p(i)x^2 + q(i)x + r(i).$$

$$p(N) = l, \qquad q(N) = w, \qquad r(N) = z.$$

6.5. Solution requires knowledge of matrix calculus and several formulas from matrix algebra, plus tedious and careful labor. We record here the recursive formulas that result, since they are useful. Defining

$$S(i) = C(i) + H^T(i)P(i + 1)H(i),$$
$$T(i) = e^T(i) + 2H^T(i)P(i + 1)k(i) + H^T(i)q^T(i + 1),$$
$$U(i) = B^T(i) + 2H^T(i)P(i + 1)G(i),$$

the recursive formulas are

$$P(i) = A(i) + G^T(i)P(i + 1)G(i) - \frac{U^T(i)S^{-1}(i)U(i)}{4},$$

$$q(i) = d(i) + 2k^T(i)P(i + 1)G(i) + q(i + 1)G(i) - \frac{T^T(i)S^{-1}(i)U(i)}{2},$$

$$r(i) = f(i) + k^T(i)P(i + 1)k(i) + q(i + 1)k(i) + r(i + 1) - \frac{T^T(i)S^{-1}(i)T(i)}{4},$$

and

$$y(i) = -\frac{S^{-1}(i)[U(i)x(i) + T(i)]}{2}.$$

$$P(N) = L, \qquad q(N) = w, \qquad r(N) = z.$$

6.6. $x(1) = \frac{1}{2}$, $x(2) = 0$. $J = .64870$.

$$J \approx \left(-\frac{7}{8}\right)^4 + 4\left(-\frac{7}{8}\right)^3\left[y(0) + \frac{7}{8}\right] + 6\left(-\frac{7}{8}\right)^2\left[y(0) + \frac{7}{8}\right]^2$$
$$+ \left(\frac{1}{2}\right)^4 + 4\left(\frac{1}{2}\right)^3\left[x(1) - \frac{1}{2}\right] + 6\left(\frac{1}{2}\right)^2\left[x(1) - \frac{1}{2}\right]^2$$
$$+ \left(-\frac{1}{16}\right)^4 + 4\left(-\frac{1}{16}\right)^3\left[y(1) + \frac{1}{16}\right] + 6\left(-\frac{1}{16}\right)^2\left[y(1) + \frac{1}{16}\right]^2.$$

$$x(1) \approx \frac{1}{2} + 4[x(0) - 2] + 4\left[y(0) + \frac{7}{8}\right].$$
$$x(2) \approx \left[x(1) - \frac{1}{2}\right] + 4\left[y(1) + \frac{1}{16}\right].$$
$$p(2) = 0, \quad q(2) = 0. \quad p(1) = \frac{3}{2}, \quad q(1) = -1.$$

$$y(0) = -\frac{\left[2\left(\frac{3}{2}\right)(4)(4)\right]x + 4\left(-\frac{7}{8}\right)^3 + 6\left(-\frac{7}{8}\right)^2\left(\frac{7}{4}\right) + 3(4)\left[\frac{1}{2} - 8 + \frac{7}{2}\right] + (-1)(4)}{2\left[6\left(\frac{7}{8}\right)^2 + \left(\frac{1}{2}\right)(4)^2\right]}$$

$$= -.86311 \quad \text{when} \quad x(0) = 2.$$

$$x(1) = .54754, \quad y(1) = -.04160, \quad x(2) = .1334. \quad J = .64516.$$

Repeating the procedure, approximating about the new solution,

$$p(2) = .106773, \quad q(2) = -.018991. \quad p(1) = 1.799573, \quad q(1) = -1.313927.$$
$$y(0) = -.863938, \quad x(1) = .544248, \quad y(1) = -.051671, \quad x(2) = .089522. \quad J = .644906.$$

The above is a successive approximation procedure as discussed in Section 1.

6.7. Since $V_i(x) = p(i)x^2 + q(i)x + r(i)$, $V_i(0) = r(i)$. $V_i(x) - V_i(-x) = 2q(i)x$, hence the result. Now assume that the problem is described by (6.1) and (6.2).

If $x(0)$ equals zero, by choosing all subsequent y's equal to zero, all subsequent x's are zero and the cost is 0. The problem cannot have a finite minimum less than zero. If it did, multiplying all y's and x's in that solution by a nonzero constant K would still satisfy (6.1) and would multiply the criterion value by K^2, giving a smaller criterion value and hence a contradiction.

Consider the optimal sequence of y's and x's starting from $x_0(i)$, a given positive value of $x(i)$. For initial point $-x_0(i)$, by reversing the signs of the y's in the previous solution, (6.1) would give the same sequence of x's as before, but with signs reversed. (6.2) would give

the same criterion value for the two solutions. The problem starting at $-x_0(i)$ cannot have a better solution, or by reversing signs of the x's and y's, we would have a better solution to the original problem, starting at $x_0(i)$, which would contradict our assumption that the solution was optimal.

6.8. Since $V_0(x)$ is the minimum cost from any state x at stage 0 to the end, the optimal choice of $x(0)$ minimizes $V_0(x)$. Hence

$$x(0) = -q(0)/2p(0),$$

assuming $p(0)$ is positive.

6.9. Define $G_i(x) =$ the minimum cost of going from stage 0 to state x at stage i. Then

$$G_i(x) = \min_{y(i-1)} \left[a(i)x^2 + c(i-1)y^2(i-1) + G_{i-1}\left(\frac{x - h(i-1)y(i-1) - k(i-1)}{g(i-1)} \right) \right].$$

$$G_0(x) = lx^2(0) + wx(0) + z.$$

Assume $G_{i-1}(x) = u(i-1)x^2 + v(i-1)x + t(i-1)$. Hence $u(0) = l$, $v(0) = w$, $t(0) = z$. Then the minimizing y is given by

$$y(i-1) = \frac{2u(i-1)h(i-1)x + [v(i-1)g(i-1)h(i-1) - 2u(i-1)h(i-1)k(i-1)]}{2[c(i-1)g^2(i-1) + u(i-1)h^2(i-1)]}.$$

Substituting, simplifying, and letting $d(i-1) = c(i-1)g^2(i-1) + u(i-1)h^2(i-1)$,

$$G_i(x) = \left[a(i) + \frac{u(i-1)c(i-1)}{d(i-1)} \right]x^2$$

$$+ \left[\frac{v(i-1)c(i-1)g(i-1) - 2u(i-1)c(i-1)k(i-1)}{d(i-1)} \right]x$$

$$+ \left[\frac{4u(i-1)c(i-1)k^2(i-1) - 4v(i-1)c(i-1)g(i-1)k(i-1) - v^2(i-1)h^2(i-1)}{4d(i-1)} \right]$$

$$+ t(i-1) \Bigg]$$

$$= u(i)x^2 + v(i)x + t(i).$$

Solve for the u, v, and t sequences. Compute $y(N-1)$ in terms of the specified $x(N)$ and $u(N-1)$ and $v(N-1)$ by the above formula for $y(i-1)$. Compute $x(N-1)$ by

$$x(N-1) = \frac{x - h(N-1)y(N-1) - k(N-1)}{g(N-1)}.$$

Now compute $y(N-2)$ by the above formula and repeat the process until $x(0)$ is found.

6.10. By the backward procedure:

$$p(2) = 1, \quad q(2) = 0, \quad r(2) = 0.$$
$$p(1) = \tfrac{3}{2}, \quad q(1) = 1, \quad r(1) = \tfrac{1}{2}.$$
$$p(0) = \tfrac{8}{3}, \quad q(0) = \tfrac{8}{3}, \quad r(0) = \tfrac{7}{3}.$$

The optimal $x(0) = -\tfrac{1}{2}$ and $V_0(-\tfrac{1}{2}) = 1$.

$$y(0) = -\tfrac{1}{2}, \quad x(1) = 0, \quad y(1) = -\tfrac{1}{2}, \quad x(2) = \tfrac{1}{2}. \quad \text{Cost} = 1.$$

By the forward procedure:

$$u(0) = 1, \quad v(0) = 0, \quad t(0) = 0.$$
$$u(1) = \tfrac{3}{2}, \quad v(1) = -1, \quad t(1) = \tfrac{1}{2}.$$
$$u(2) = \tfrac{8}{3}, \quad v(2) = -\tfrac{8}{3}, \quad t(2) = \tfrac{7}{3}.$$

The optimal $x(2) = \frac{1}{2}$ and $G_2(\frac{1}{2}) = 1$. $y(1) = -\frac{1}{2}$, hence $x(1) = 0$. $y(0) = -\frac{1}{2}$, hence $x(0) = -\frac{1}{2}$. Cost $= 1$.

6.11.

$$F_k(x) = (z(k) - x)^2 + F_{k-1}\left(\frac{x - d(k-1)}{b(k-1)}\right).$$

$$F_0(x) = (z(0) - x)^2 = x^2 - 2z(0)x + z^2(0).$$

Assume $F_{k-1}(x) = p(k-1)x^2 + q(k-1)x + r(k-1)$.

$$F_k(x) = (z(k) - x)^2 + p(k-1)\left(\frac{x - d(k-1)}{b(k-1)}\right)^2 + q(k-1)\left(\frac{x - d(k-1)}{b(k-1)}\right) + r(k-1)$$

$$= \left[1 + \frac{p(k-1)}{b^2(k-1)}\right]x^2 + \left[\frac{q(k-1)}{b(k-1)} - \frac{2p(k-1)d(k-1)}{b^2(k-1)} - 2z(k)\right]x$$

$$+ \left[r(k-1) - z^2(k) + \frac{p(k-1)d^2(k-1)}{b^2(k-1)} - \frac{q(k-1)d(k-1)}{b(k-1)}\right]$$

$$= p(k)x^2 + q(k)x + r(k).$$

The above are recurrence relations for p, q, and r and $p(0) = 1$, $q(0) = -2z(0)$, $r(0) = z^2(0)$. If $z(i)$, $i = 0, \ldots, k$, are given, $\tilde{x}(k)$, the minimum squared-error estimate of $x(k)$, is $-q(k)/2p(k)$. Using the above formulas, one can show

$$\tilde{x}(k+1) = b(k)\tilde{x}(k) + d(k) + K(k)[z(k+1) - b(k)\tilde{x}(k) - d(k)],$$

where $K(k) = b^2(k)/[p(k) + b^2(k)]$. This says that the estimated state at stage $k + 1$ is the state given by plugging the estimated state at stage k into the dynamics, plus a weight $K(k)$ times the difference between the observation at stage $k + 1$ and what one would expect based on the estimated state at stage k. Note that $K(k)$ depends only on $p(k)$ and that $p(i)$, $i = 0, 1, \ldots$, does not depend on the observations. Hence $K(i)$, $i = 0, 1, \ldots$, can be computed before any observations and the estimated state at stage $k + 1$ can be computed quite easily when $z(k + 1)$ is observed assuming that $\tilde{x}(k)$ is already known.

This is a very important sequential estimation, or filtering, procedure, sometimes called Kalman filtering. It has proved useful in updating estimated rocket positions based on real-time, error-corrupted observations.

6.12. Define

$V_i(x_1, x_2) = $ the minimum remaining cost for a process starting stage i with current state $x(i)$ equal to x_1 and previous state $x(i - 1)$ equal to x_2.

$$V_i(x_1, x_2) = \min_y [a(i)x_1^2 + c(i)y^2 + V_{i+1}(g_1(i)x_1 + g_2(i)x_2 + h(i)y, x_1)].$$

$$V_N(x_1, x_2) = lx_1^2.$$

$$V_{N-1}(x_1, x_2) = \min_y \left\{ a(N-1)x_1^2 + c(N-1)y^2 \right.$$

$$\left. + l[g_1(N-1)x_1 + g_2(N-1)x_2 + h(N-1)y]^2 \right\}.$$

$$c(N-1)y + l[g_1(N-1)x_1 + g_2(N-1)x_2 + h(N-1)y]h(N-1) = 0.$$

$$y = -\frac{lg_1(N-1)h(N-1)x_1 + lg_2(N-1)h(N-1)x_2}{c(N-1) + lh^2(N-1)}.$$

Substitution shows $V_{N-1}(x_1, x_2)$ to be of the form $p_{11}(N-1)x_1^2 + p_{12}(N-1)x_1x_2 +$

$p_{22}(N-1)x_2^2$. Assuming that $V_{i+1}(x_1, x_2)$ is of this form, it can be verified that $V_i(x_1, x_2)$ is also. The optimal decision y is linear in x_1 and x_2. The answer is $V_1(c_1, c_0)$.

6.13. Define

$F_i(x, y_1) =$ the minimum remaining cost for a process starting stage i with current state $x(i)$ equal to x and previous decision $y(i-1)$ equal to y_1.

$$F_i(x, y_1) = \min_y \left[a(i)x^2 + c(i)y^2 + d(i)(y - y_1)^2 + F_{i+1}(g(i)x + h(i)y, y) \right].$$

Assume that $F_{i+1}(x, y_1)$ is of the form $u_{11}(i+1)x^2 + u_{12}(i+1)xy_1 + u_{22}(i+1)y_1^2$ and obtain recurrence relations for u_{11}, u_{12}, and u_{22}. The optimal decision is linear in x and y_1. The answer is $F_1(g(0)x(0) + h(0)y(0), y(0))$.

6.14. The optimal value function is neither linear nor quadratic and no procedure of the sort given in the text can be used to solve the problem.

6.15. Define $V_i(x)$ as in the text.

$$V_N(x) = lx^2.$$

$$V_{N-1}(x) = \min_{-1 < y < 1} \left\{ a(N-1)x^2 + c(N-1)y^2 + l[g(N-1)x + h(N-1)y]^2 \right\}.$$

Setting the derivative with respect to y equal to 0,

$$y = -\frac{lg(N-1)h(N-1)x}{c(N-1) + lh^2(N-1)} = -s(N-1)x.$$

If $s(N-1)$ is positive, y is given by the above formula for

$$-1/s(N-1) < x < 1/s(N-1).$$

For $x < -1/s(N-1)$, $y = 1$. For $x > 1/s(N-1)$, $y = -1$. Substituting, $V_{N-1}(x)$ is given by (6.10) for $-1/s(N-1) < x < 1/s(N-1)$. For $x < -1/s(N-1)$,

$$V_{N-1}(x) = a(N-1)x^2 + c(N-1) + l[g(N-1)x + h(N-1)]^2$$

which is quadratic in x. Similarly for $x > 1/s(N-1)$, where $y = -1$. Hence $V_{n-1}(x)$ is piecewise quadratic, with different equations in three different regions. $V_{N-2}(x)$ is likewise piecewise quadratic, with up to nine different equations. Similarly for all $V_i(x)$.

No general method is known for recursively computing the appropriate coefficients and the boundaries of the associated regions.

6.16. $V_2(x) = 6x^2 - 4x + 1$ by (6.22). $p(1) = 300/7$ and $q(1) = -48/7$ by result of Problem 6.4. Hence $y(0) = -3$ by result of Problem 6.4. By the dynamics, $x(1) = \frac{1}{2}$. By Problem 6.4, $y(1) = -1$. By dynamics, $x(2) = 1$. By (6.21), $y(2) = -1$ and by dynamics $x(3) = 2$.

6.17.

$$x_1(3) = 2 = 2x_1(1) + 3y(1) + 2y(2).$$
$$x_2(3) = 1 = 2x_2(1) + y(1) + y(2).$$

Hence $y(1) = -2x_1(1) + 4x_2(1)$. $y(2) = 1 + 2x_1(1) - 6x_2(1)$. Therefore $x_1(2) = -3x_1(1) + 9x_2(1)$ and $x_2(2) = -x_1(1) + 3x_2(1)$.

$$V_1(x_1, x_2) = x_1^2 + x_2^2 + (-2x_1 + 4x_2)^2 + (-3x_1 + 9x_2)^2 + (-x_1 + 3x_2)^2 + (1 + 2x_1 - 6x_2)^2$$
$$= 19x_1^2 - 100x_1x_2 + 143x_2^2 + 4x_1 - 12x_2 + 1.$$

$$P(1) = \begin{bmatrix} 19 & -50 \\ -50 & 143 \end{bmatrix}, \qquad q(1) = [4 \ \ -12], \qquad r(1) = 1.$$

$$C(0) = 1, \qquad G(0) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad H(0) = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

all other data are zero. Hence, $S(0) = 20$, $T(0) = -4$, $U(0) = [62 \ -110]$. $y(0) = -8$.

$$P(0) = \frac{1}{20}\begin{bmatrix} 279 & -775 \\ -775 & 2215 \end{bmatrix}, \qquad q(0) = \tfrac{1}{5}[-9 \ \ 25], \qquad r(0) = \tfrac{4}{5}.$$

$V_0(7, 1) = 245$. Checking, $y(0) = -8$ implies $x_1(1) = -8$, $x_2(1) = -2$, $y(1) = 8$, $x_1(2) = 6$, $x_2(2) = 2$, $y(2) = -3$, $x_1(3) = 2$, $x_2(3) = 1$. Evaluating J for this trajectory gives $J = 245$.

6.18. By (6.31) and (6.28), $p(N - 1) = a(N - 1)$, $q(N - 1) = g(N - 1)$,

$$r(N - 1) = -\frac{h^2(N - 1)}{4c(N - 1)}.$$

By (6.40),

$$\lambda = -\frac{2c(N - 1)(t - g(N - 1)x)}{h^2(N - 1)}.$$

By (6.30), $y(N - 1) = [t - g(N - 1)x]/h(N - 1)$, which checks with (6.21).

$$W_{N-1}(x, \lambda) = p(N - 1)x^2 + q(N - 1)\lambda x + r(N - 1)\lambda^2$$

$$= a(N - 1)x^2 + g(N - 1)\lambda x - \frac{h^2(N - 1)}{4c(N - 1)}\lambda^2.$$

Substituting λ given above and subtracting λt yields (6.22).

6.19.

$$W_i(x, \lambda) = \underset{y}{\mathrm{stat}}\left[x^T A(i)x + y^T C(i)y + (G(i)x + H(i)y)^T P(i + 1)(G(i)x + H(i)y) \right.$$

$$\left. + (G(i)x + H(i)y)^T Q(i + 1)\lambda + \lambda^T R(i + 1)\lambda \right].$$

Differentiation yields

$$y(i) = -\tfrac{1}{2}S^{-1}(i)[2H^T(i)P(i + 1)G(i)x(i) + H^T(i)Q(i + 1)\lambda],$$

where $S(i) = C(i) + H^T(i)P(i + 1)H(i)$. Substitution for $y(i)$ yields

$$P(i) = A(i) + G^T(i)P(i + 1)G(i) - G^T(i)P(i + 1)H(i)S^{-1}(i)H^T(i)P(i + 1)G(i),$$

$$Q(i) = [G^T(i) - G^T(i)P(i + 1)H(i)S^{-1}(i)H^T(i)]Q(i + 1),$$

$$R(i) = -\tfrac{1}{4}[Q^T(i + 1)H(i)S^{-1}(i)H^T(i)Q(i + 1)] + R(i + 1),$$

and $P(N) = 0$, $Q(N) = I$, $R(N) = 0$.

Now let $T_i(x, \lambda)$, an n-vector valued function of the n-vectors x and λ, be the terminal state $x(N)$ if the process starts stage i in state x for given λ and $y(i)$ as given above is used.

$$T_i(x, \lambda) = T_{i+1}(Gx + Hy, \lambda).$$

Assume $T_{i+1}(x, \lambda) = B(i + 1)x + D(i + 1)\lambda$. Clearly $B(N) = I$, $D(N) = 0$. Substitution for y yields

$$B(i) = B(i + 1)[G(i) - H(i)S^{-1}(i)H^T(i)P(i + 1)G(i)].$$

Hence $B^T(i) = [G^T(i) - G^T(i)P(i + 1)H(i)S^{-1}(i)H^T(i)]B^T(i + 1)$, so $B^T(i) = Q(i)$.

$$D(i) = -\tfrac{1}{2}[Q^T(i + 1)H(i)S^{-1}(i)H^T(i)Q(i + 1)] + D(i + 1),$$

so $D(i) = 2R(i)$. $x(N) = Q^T(i)x(i) + 2R(i)\lambda$, so $\lambda = \tfrac{1}{2}R^{-1}(i)(z - Q^T(i)x(i))$, where $z = x(N)$.

6.20. The solution of Problem 6.19 is modified as follows: Let λ be an $e \times 1$ vector, and add $(Ex)^T\lambda$ to the criterion rather than $x^T\lambda$. Now P is $n \times n$, Q is $n \times e$ and R is $e \times e$ and $Q(N) = E^T$. The recursive formulas for P, Q, and R are unaffected. T is now the terminal value of $Ex(N)$ and is an e-vector valued function. Nothing further changes, so λ is given by the same formula as in Problem 6.19, where z is now $Ex(N)$. Note that the smaller is e (the number of terminal conditions), the less is the computation required.

6.21.

$$A(1) = A(2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

$$C(1) = C(2) = 1.$$

$$G(1) = G(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

$$H(1) = H(2) = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

$$P(3) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad Q(3) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R(3) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

$$S(2) = 1.$$

$$P(2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad Q(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad R(2) = -\frac{1}{4}\begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}.$$

$$S(1) = 6.$$

$$P(1) = \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{17}{6} \end{bmatrix}, \qquad Q(1) = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{11}{6} \end{bmatrix}, \qquad R(1) = -\frac{1}{24}\begin{bmatrix} 33 & 15 \\ 15 & 7 \end{bmatrix}.$$

$$\lambda = \begin{bmatrix} -14 & 30 \\ 30 & -66 \end{bmatrix}\left[\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{11}{6} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right].$$

$$V_1(x_1, x_2) = x^\mathsf{T}P(1)x + x^\mathsf{T}Q(1)\lambda + \lambda^\mathsf{T}R(1)\lambda - [2 \quad 1]\lambda$$
$$= 19x_1^2 - 100x_1x_2 + 143x_2^2 + 4x_1 - 12x_2 + 1.$$

Since there are two state variables and one decision variable, no decision leads to the terminal point from most points at stage 2. This is reflected in the fact that $R(2)$ is singular so λ cannot be computed at stage 2.

6.22.

$$y(1) = -x(1)$$
$$V_1(x) = \tfrac{1}{2}x^2$$
$$V_0(x) = \min_{y(0)} \left[y^2(0) + \tfrac{1}{2}(x + y(0))^2 \right]$$
$$y(0) = -x(0)/3 = -\tfrac{1}{3}, \qquad y(1) = -\tfrac{2}{3}.$$

By the general method:

$$W_1(x, \lambda) = \operatorname*{stat}_{y(1)}\left[x^2 - \tfrac{1}{2}y^2(1) + \lambda(x + y(1)) \right]$$
$$= x^2 + \lambda x + \tfrac{1}{2}\lambda^2 \quad \text{where} \quad y(1) = \lambda \; maximizes$$
$$\text{the bracketed expression.}$$

$$W_0(x, \lambda) = \operatorname*{stat}_{y(0)}\left[y^2(0) + (x + y(0))^2 + \lambda(x + y(0)) + \tfrac{1}{2}\lambda^2 \right]$$
$$= \tfrac{1}{2}x^2 + \tfrac{1}{2}\lambda x + \tfrac{3}{8}\lambda^2 \quad \text{where} \quad y(0) = -(2x + \lambda)/4 \; \text{minimizes.}$$

$$\lambda = -2x(0)/3 = -\tfrac{2}{3}. \qquad y(0) = -\tfrac{1}{3}, \qquad y(1) = -\tfrac{2}{3}.$$

6.23.

$$u(2) = 6, \qquad v(2) = -2, \qquad w(2) = \tfrac{1}{4}.$$
$$u(1) = \tfrac{300}{7}, \qquad v(1) = -\tfrac{24}{7}, \qquad w(1) = \tfrac{5}{28}.$$
$$u(0) = \tfrac{75}{7} - \tfrac{1875}{322}, \qquad v(0) = -\tfrac{12}{7} + \tfrac{300}{322}, \qquad w(0) = \tfrac{5}{28} - \tfrac{12}{322}.$$

$F_0(2, 2) = 17$ which is 18, the cost of the optimal solution as given in Problem 6.3, minus 1, which is $\frac{1}{4}x^2(3)$ as given by the optimal solution. $y(0)$, by (6.47), is -3 and, repeatedly using (6.47), we get the same trajectory as previously.

$F_0(4, -19) = 188\frac{3}{4}$. Using (6.47) with the appropriate $x(i)$ and $z = -19$, gives $y(0) = -9$, $x(1) = \frac{1}{2}$, $y(1) = -4$, $x(2) = -\frac{1}{2}$, $y(2) = -\frac{17}{2}$, and hence J by (6.3) neglecting $\frac{1}{4}x^2(3)$ is $188\frac{3}{4}$.

6.24. Total cost from given $x(0)$ to given $x(N)$ using a minimum cost connecting solution is

$$\text{total cost} = 2x^2(0) + u(0)x^2(0) + v(0)x(0)x(N) + w(0)x^2(N) + \frac{1}{4}x^2(N)$$

where $u(0)$, $v(0)$, and $w(0)$ are calculated in Problem 6.23. We could use the conventional Lagrange multiplier method to minimize total cost subject to $x(0) + 2x(N) = 1$, or, solving the constraint for $x(0)$ and substituting

$$\text{cost} = (2 + u(0))(1 - 2x(N))^2 + v(0)(1 - 2x(N))x(N) + (w(0) + \frac{1}{4})x^2(N).$$

Differentiating with respect to $x(N)$ to minimize gives

$$0 = 2(2 + u(0))(1 - 2x(N))(-2) + v(0)(1 - 2x(N)) + v(0)x(N)(-2) + 2(w(0) + \frac{1}{4})x(N).$$

This linear equation can be solved for $x(N)$ and then the constraint determines $x(0)$ and the solution of Problem 6.23 yields the connecting sequence.

6.25. $F_{2i}(x, z) = \min_t[F_i(x, t) + F_i(t, z)]$ where t is the choice of state at stage i of a process of duration $2i$.

$$F_1(x, z) = ax^2 + c\left[\frac{z - gx}{h}\right]^2 = \left(a + \frac{cg^2}{h^2}\right)x^2 - \frac{2cg}{h^2}xz + \frac{c}{h^2}z^2 = u(1)x^2 + v(1)xz + w(1)z^2.$$

Assume $F_i(x, z) = u(i)x^2 + v(i)xz + w(i)z^2$. Setting the derivative with respect to t equal to 0 gives

$$t = -\frac{v(i)}{2[u(i) + w(i)]}(x + z).$$

Substitution for t yields

$$F_{2i}(x, z) = \left\{u(i) - \frac{v^2(i)}{4[u(i) + w(i)]}\right\}x^2 + \left\{-\frac{v^2(i)}{2[u(i) + w(i)]}\right\}xz$$

$$+ \left\{w(i) - \frac{v^2(i)}{4(u(i) + w(i))}\right\}z^2$$

which gives recurrence relations for $u(2i)$, $v(2i)$, and $w(2i)$.

Chapter 7

7.1. If $x(N)$ is specified, the two derivations are modified as follows:

Derivation 1. (7.8) is unchanged since each of $x(1), \ldots, x(N-1)$ can still be varied, with the other x's held fixed. (7.9) cannot be written since $x(N)$ cannot vary. This condition is replaced by the condition that $x(N)$ equals its specified value.

Derivation 2. $S_N(x)$ is defined only at one point. $S_{N-1}(x) = g_{N-1}(x(N-1), y(N-1))$ where $y(N-1)$ is given, in terms of $x(N-1)$ and the specified $x(N)$, by

$$x(N) = x(N-1) + f_{N-1}(x(N-1), y(N-1)). \tag{S7.1}$$

(7.13) holds for $i = 0, \ldots, N-2$ and (7.12) holds for $i = 0, \ldots, N-2$. These results combined yield (7.8) for $i = 1, \ldots, N-2$. By partial differentiation of $S_{N-1}(x)$ we obtain

$$\frac{\partial S_{N-1}}{\partial x(N-1)} = \frac{\partial g_{N-1}}{\partial x(N-1)} + \frac{\partial g_{N-1}}{\partial y(N-1)} + \frac{\partial y(N-1)}{\partial x(N-1)}. \tag{S7.2}$$

Partial differentiation of (S7.1) with respect to $x(N-1)$ yields

$$0 = 1 + \frac{\partial f_{N-1}}{\partial x(N-1)} + \frac{\partial f_{N-1}}{\partial y(N-1)} \frac{\partial y(N-1)}{\partial x(N-1)} . \tag{S7.3}$$

Substitution in (S7.2) of $\partial y(N-1)/\partial x(N-1)$ given by (S7.3) and of $\partial S_{N-1}/\partial x(n-1)$ given by (7.12) for $i = N-2$ gives (7.8) for $i = N-1$.

If $x(0)$ is unspecified and a cost of $k(x(0))$ is added, the derivations are modified as follows:

Derivation 1. $\dot{x}(0)$ can now be varied yielding

$$\frac{\partial k(x(0))}{\partial x(0)} + \frac{\partial g_0}{\partial x(0)} + \frac{\partial g_0}{\partial y(0)} \frac{\partial y(0)}{\partial x(0)} = 0,$$

hence

$$\frac{\partial k(x(0))}{\partial x(0)} + \frac{\partial g_0}{\partial x(0)} - \frac{\partial g_0/\partial y(0)}{\partial f_0/\partial y(0)} \left(1 + \frac{\partial f_0}{\partial x(0)} \right) = 0. \tag{S7.4}$$

This relationship between $x(0)$ and $y(0)$ replaces the fact, in the original problem, that $x(0)$ was specified.

Derivation 2. (7.13) holds for $i = 0$ and $\partial S_0/\partial x(0) = -\partial k(x(0))/\partial x(0)$ since $x(0)$ minimizes $S_0(x(0)) + k(x(0))$. Hence result (S7.4).

7.2. Note the different definitions of the dynamics in the two chapters: f_i in this chapter equals $[g(i) - 1]x(i) + h(i)y(i)$ in Chapter 6.

$$\partial V_0(x)/\partial x = 9x(0) = 18 = \partial S_0/\partial x(0).$$

$$2y(0) + \frac{\partial S_1}{\partial x(1)} \cdot \frac{1}{6} = 0, \qquad \frac{\partial S_0}{\partial x(0)} = 18 = \frac{\partial S_1}{\partial x(1)} \left(1 - \frac{1}{2} \right).$$

Hence $y(0) = -3$, $\partial S_1/\partial x(1) = 36$. $x(1) = \frac{1}{2}$.

$$4y(1) + \frac{\partial S_2}{\partial x(2)} \cdot \frac{1}{2} = 0, \qquad \frac{\partial S_1}{\partial x(1)} = 36 = 24 \cdot \frac{1}{2} + \frac{\partial S_2}{\partial x(2)} \cdot 3.$$

Hence $y(1) = -1$, $\partial S_2/\partial x(2) = 8$. $x(2) = 1$.

$$2y(2) + \frac{\partial S_3}{\partial x(3)} \cdot 2 = 0, \qquad \frac{\partial S_2}{\partial x(2)} = 8 = 4 + \frac{\partial S_3}{\partial x(3)} \cdot 4.$$

Hence $y(2) = -1$, $\partial S_3/\partial x(3) = 1$. $x(3) = 2$.

Checking (7.14), $1 \overset{?}{=} \frac{1}{2} \cdot 2 = 1$. It checks.

Using the results of derivation 1, "guess" $y(0) = -3$. $x(1) = \frac{1}{2}$.

$$\frac{-6}{\frac{1}{6}} + 12 - 4y(1) \frac{3}{\frac{1}{2}} = 0.$$

Hence $y(1) = -1$, $x(2) = 1$.

$$\frac{-4}{\frac{1}{2}} + 4 - 2y(2) \frac{4}{2} = 0.$$

Hence $y(2) = -1$, $x(3) = 2$.

Checking (7.9), $-2/2 + 1 \overset{?}{=} 0$. It checks.

7.3. (7.3) becomes

$$\frac{\partial g_{(i-1)\Delta}}{\partial y((i-1)\Delta)} \Delta \frac{\partial y((i-1)\Delta)}{\partial x(i\Delta)} + \frac{\partial g_{i\Delta}}{\partial x(i\Delta)} \Delta + \frac{\partial g_{i\Delta}}{\partial y(i\Delta)} \Delta \frac{\partial y(i\Delta)}{\partial x(i\Delta)} = 0 \quad (i = 1, \ldots, N-1),$$

$$\frac{\partial g_{(N-1)\Delta}}{\partial y((N-1)\Delta)} \Delta \frac{\partial y((N-1)\Delta)}{\partial x(N\Delta)} + \frac{\partial h}{\partial x(N\Delta)} = 0.$$

(7.6) becomes $\partial y((i-1)\Delta)/\partial x(i\Delta) = 1/\Delta$, and (7.7) becomes $\partial y(i\Delta)/\partial x(i\Delta) = -1/\Delta$. Hence (7.8) becomes

$$\frac{\partial g_{(i-1)\Delta}}{\partial y((i-1)\Delta)} + \frac{\partial g_{i\Delta}}{\partial x(i\Delta)} \Delta - \frac{\partial g_{i\Delta}}{\partial y(i\Delta)} = 0 \qquad (i = 1, \ldots, N-1). \qquad (S7.5)$$

(7.9) becomes

$$\frac{\partial g_{(N-1)\Delta}}{\partial y((N-1)\Delta)} + \frac{\partial h}{\partial x(N\Delta)} = 0.$$

As $\Delta \to 0$ and $N \to \infty$ so that $N\Delta = T$, (S7.5) becomes

$$\lim_{\Delta \to 0} \frac{1}{\Delta} \left(\frac{\partial g_{i\Delta}}{\partial y(i\Delta)} - \frac{\partial g_{(i-1)\Delta}}{\partial y((i-1)\Delta)} \right) = \frac{\partial g_{i\Delta}}{\partial x(i\Delta)}$$

or, by the definition of the derivative and the fact that $y(i\Delta) \to \dot{x}(t)$

$$\frac{d}{dt} \frac{\partial g_t}{\partial \dot{x}(t)} = \frac{\partial g_t}{\partial x(t)}. \qquad (S7.6)$$

This is called the Euler–Lagrange differential equation necessary condition for the calculus of variations problem.

7.4. (7.8) becomes

$$\frac{y(i-1)}{\sqrt{1 + y^2(i-1)}} - \frac{y(i)}{\sqrt{1 + y^2(i)}} = 0. \qquad (S7.7)$$

Hence, squaring,

$$\frac{y^2(i-1)}{1 + y^2(i-1)} = \frac{y^2(i)}{1 + y^2(i)}$$

which implies $y(i-1) = y(i)$ since $y(i-1)$ and $y(i)$ are of the same sign by (S7.7). Hence, from the dynamics,

$$x(i) - x(i-1) = x(i+1) - x(i)$$

so x increases by the same amount at each stage, which means the points $x(i)$ lie on a straight line.

Assuming that this unique stationary solution is a minimum, we have shown that a straight line is the shortest path consisting of linear segments between two points. A similar argument based on result (S7.6) (with $g_t = \sqrt{1 + \dot{x}^2}$) of the previous problem shows that a straight line is a stationary (minimum) solution to the shortest distance problem.

7.5. *Derivation 1.* Regarding $x(0), \ldots, x(N-1)$ as independent, and $x(N)$ as dependent on $x(0)$ through

$$x(0) + x(N) = 1, \qquad (S7.8)$$

we get, taking the partial derivative of J with respect to $x(i)$, $i = 1, \ldots, N-1$,

$$\frac{\partial g_{i-1}}{\partial x(i)} + \frac{\partial g_i}{\partial x(i)} = 0 \qquad (i = 1, \ldots, N-1), \qquad (S7.9)$$

and, taking the partial derivative of J with respect to $x(0)$,

$$\frac{\partial g_0}{\partial x(0)} + \frac{\partial g_{N-1}}{\partial x(N)} \frac{\partial x(N)}{\partial x(0)} = 0 = \frac{\partial g_0}{\partial x(0)} - \frac{\partial g_{N-1}}{\partial x(N)}. \qquad (S7.10)$$

(S7.8)–(S7.10) are $N+1$ equations for the $N+1$ unknown x's.

Derivation 2. Define $S_i(x(i), x(N)) =$ the minimum cost for a process starting stage i in state $x(i)$ and terminating at $x(N)$.

$$S_i(x(i), x(N)) = \min_{x(i+1)} [g_i(x(i), x(i+1)) + S_{i+1}(x(i+1), x(N))] \quad (i = 0, \ldots, N-2).$$

$$\tag{S7.11}$$

$$S_{N-1}(x(N-1), x(N)) = g_{N-1}(x(N-1), x(N)). \tag{S7.12}$$

Minimization yields

$$\frac{\partial g_i}{\partial x(i+1)} + \frac{\partial S_{i+1}}{\partial x(i+1)} = 0 \quad (i = 0, \ldots, N-2). \tag{S7.13}$$

Partial differentiation of (S7.11), using (S7.13) to eliminate $\partial x(i+1)/\partial x(i)$ and $\partial x(i+1)/\partial x(N)$ terms, yields

$$\partial S_i/\partial x(i) = \partial g_i/\partial x(i) \quad (i = 0, \ldots, N-2), \tag{S7.14}$$

$$\partial S_i/\partial x(N) = \partial S_{i+1}/\partial x(N) \quad (i = 0, \ldots, N-2). \tag{S7.15}$$

(S7.12) implies

$$\partial S_{N-1}/\partial x(N-1) = \partial g_{N-1}/\partial x(N-1), \tag{S7.16}$$

$$\partial S_{N-1}/\partial x(N) = \partial g_{N-1}/\partial x(N). \tag{S7.17}$$

Finally, the fact that $x(0)$ minimizes S_0 (with $x(N)$ dependent on $x(0)$ through $x(0) + x(N) = 1$) yields

$$\frac{\partial S_0}{\partial x(0)} + \frac{\partial S_0}{\partial x(N)} \frac{\partial x(N)}{\partial x(0)} = 0 = \frac{\partial S_0}{\partial x(0)} - \frac{\partial S_0}{\partial x(N)}. \tag{S7.18}$$

Results (S7.14)–(S7.18) combined duplicate the results of derivation 1.

While derivation 1 is simpler than derivation 2 for the problems of this section, derivation 2 is important because only it generalizes to the multidimensional problem that is studied in Section 3.

7.6. *Derivation 1.* Observe that since $x(j)$ appears in the dynamics for $i = j-1$, j, and $j+1$, changing $x(j)$ while holding all other x's fixed requires a compensating change in $y(j-1)$, $y(j)$, and $y(j+1)$. Hence

$$\frac{\partial g_{j-1}}{\partial y(j-1)} \frac{\partial y(j-1)}{\partial x(j)} + \frac{\partial g_j}{\partial x(j)} + \frac{\partial g_j}{\partial y(j)} \frac{\partial y(j)}{\partial x(j)} + \frac{\partial g_{j+1}}{\partial y(j+1)} \frac{\partial y(j+1)}{\partial x(j)} = 0$$

$$(j = 2, \ldots, N-2). \quad (S7.19)$$

By partial differentiation of the dynamics, for $j = 2, \ldots, N-2$,

$$\frac{\partial y(j-1)}{\partial x(j)} = \frac{1}{\partial f_{j-1}/\partial y(j-1)}, \quad \frac{\partial y(j)}{\partial x(j)} = -\frac{\partial f_j/\partial x(j)}{\partial f_j/\partial y(j)}, \quad \frac{\partial y(j+1)}{\partial x(j)} = -\frac{\partial f_{j+1}/\partial x(j)}{\partial f_{j+1}/\partial y(j+1)}.$$

$$\tag{S7.20}$$

Substitution of (S7.20) in (S7.19) gives $N-3$ equations. For $j = N-1$ we get (S7.20) with the last term omitted, and for $j = N$ we get

$$\frac{\partial g_{N-1}/\partial y(N-1)}{\partial f_{N-1}/\partial y(N-1)} + \frac{\partial h}{\partial x(N)} = 0. \tag{S7.21}$$

These are $N-1$ equation for the $N-1$ unknown x's. By guessing $y(1)$ and $y(2)$ and using the dynamics to obtain $x(2)$ and $x(3)$, (S7.19) and (S7.20) with $j = 2$ yields $y(3)$ in terms of known quantities. This process may be continued until $x(N)$ is found using (S7.19) with $j = N-2$. $y(1)$ and $y(2)$ must be guessed in such a way that (S7.19) with $j = N-1$ (and the last term omitted) and (S7.21) are satisfied.

Derivation 2. Define $S_i(x(i), x(i-1)) =$ the minimum cost for a process starting stage i in state $x(i)$ with previous state $x(i-1)$.

$$S_i(x(i), x(i-1)) = \min_{y(i)} \left[g_i(x(i), y(i)) + S_{i+1}(f_i(x(i), x(i-1), y(i)), x(i)) \right]$$
$$(i = 1, \ldots, N-1),$$
$$S_N(x(N), x(N-1)) = h(x(N)).$$

From this we get the three results

$$\frac{\partial g_i}{\partial y(i)} + \frac{\partial S_{i+1}}{\partial x(i+1)} \frac{\partial f_i}{\partial y(i)} = 0 \qquad (i = 1, \ldots, N-1), \qquad \text{(S7.22)}$$

$$\frac{\partial S_i}{\partial x(i)} = \frac{\partial g_i}{\partial x(i)} + \frac{\partial S_{i+1}}{\partial x(i+1)} \frac{\partial f_i}{\partial x(i)} + \frac{\partial S_{i+1}}{\partial x(i)} = 0 \qquad (i = 2, \ldots, N-1), \qquad \text{(S7.23)}$$

$$\frac{\partial S_i}{\partial x(i-1)} = \frac{\partial S_{i+1}}{\partial x(i+1)} \frac{\partial f_i}{\partial x(i-1)} \qquad (i = 3, \ldots, N-1).$$

These results yield (S7.19) with (S7.20) substituted. From $\partial S_N/\partial x(N) = \partial h/\partial x(N)$ and (S7.22) with $i = N-1$ we get (S7.21) and from $\partial S_N/\partial x(N-1) = 0$ and (S7.23) with $i = N-1$ and (S7.22) we get the last condition.

7.7.

$$\frac{\partial V_0}{\partial x(0)} = 2P(0)x(0) + q^T(0) = \begin{bmatrix} 116 \\ -316 \end{bmatrix} = \begin{bmatrix} \partial S_0/\partial x_1(0) \\ \partial S_0/\partial x_2(0) \end{bmatrix}.$$

$$116 = \frac{\partial S_1}{\partial x_1(1)} + \frac{\partial S_1}{\partial x_2(1)}, \qquad -316 = \frac{\partial S_1}{\partial x_1(1)} - \frac{\partial S_1}{\partial x_2(1)}, \qquad 0 = 2y(0) + 2\frac{\partial S_1}{\partial x_1(1)} + \frac{\partial S_1}{\partial x_2(1)}.$$

Solving, $\partial S_1/\partial x_1(1) = -100$, $\partial S_1/\partial x_2(1) = 216$, $y(0) = -8$, $x_1(1) = -8$, $x_2(1) = -2$.

The above process can be repeated to complete the solution, but $\partial S_2/\partial x$ and $\partial S_3/\partial x$ cannot be interpreted as partial derivatives of the usual optimal value function. The derivation uses multipliers to convert to a free terminal point problem (as in Chapter 6) but is beyond the scope of this book.

7.8.

$$\delta f = \frac{\partial f}{\partial x_1} \delta x_1 + \frac{\partial f}{\partial x_2} \delta x_2.$$

Let $\delta x_1 = k \,\partial f/\partial x_1$ and $\delta x_2 = k \,\partial f/\partial x_2$.

$$\delta f = k \left[\left(\frac{\partial f}{\partial x_1} \right)^2 + \left(\frac{\partial f}{\partial x_2} \right)^2 \right]$$

so

$$k = \frac{\delta f}{(\partial f/\partial x_1)^2 + (\partial f/\partial x_2)^2}$$

and

$$\delta x_1 = \frac{(\delta f)\partial f/\partial x_1}{(\partial f/\partial x_1)^2 + (\partial f/\partial x_2)^2}, \qquad \delta x_2 = \frac{(\delta f)\partial f/\partial x_2}{(\partial f/\partial x_1)^2 + (\partial f/\partial x_2)^2}.$$

$\partial f/\partial x_1|_{1,1} = 2$, $\partial f/\partial x_2|_{1,1} = 4$, so $\delta x_1 = -.3$, $\delta x_2 = -.6$ and $f(.7, .4) = .81$. This is an improvement over $f(1, 1) = 3$, but a lesser improvement than sought. Now

$$\frac{\partial f}{\partial x_1}\bigg|_{.7, .4} = 1.4, \qquad \frac{\partial f}{\partial x_2}\bigg|_{.7, .4} = 1.6. \qquad \delta x_1 = -.93, \qquad \delta x_2 = -1.06.$$

$x_1 = -.23$, $x_2 = -.66$, $f(-.23, -.66) = .92$ so let $\delta f = -\frac{3}{2}$. Then

$$\delta x_1 = -.47, \qquad \delta x_2 = -.53, \qquad x_1 = .23, \qquad x_2 = -.13, \qquad f(.23, -.13) = .09.$$

7.9. $\partial T_2/\partial x(2) = 0$, $\partial T_1/\partial x(1) = \frac{1}{2}$; hence, $\partial T_1/\partial y(1) = -.00098$, $\partial T_0/\partial y(0) = -.67969$. $k = -.01015$; hence $\delta y(0) = .00690$, $\delta y(1) = .00001$. The new solution is $y(0) = -.8681$, $x(1) = .5276$, $y(1) = -.06249$, $x(2) = .02840$. $J = .64541$ which is an improvement over .64870 of the original solution but the reduction in J of .00329 is less than the .005 sought. Applying formulas (7.27) and (7.26) to this new solution (noting that due to the form of (7.2), $f_i = x^2(i) - x(i) + 4y(i)$):

$$\partial T_2/\partial x(2) = .00002, \qquad \partial T_1/\partial x(1) = .58748;$$

hence $\partial T_1/\partial y(1) = -.00016$, $\partial T_0/\partial y(0) = -.26687$.

$k = -.07020$; hence $\delta y(0) = .01874$, $\delta y(1) = .00001$. The new solution is $y(0) = -.84936$, $x(1) = .60256$, $y(1) = -.06248$, $x(2) = .11316$. $J = .65245$ which exceeds .64541, so reject this solution and divide δJ, hence k, by 2.

$k = -.03510$; hence $\delta y(0) = .00937$, $\delta y(1) = .00001$. The new solution is $y(0) = -.85873$, $x(1) = .56508$, $y(1) = -.06248$, $x(2) = .06940$. $J = .64578$ which still exceeds .64541, so reject and again divide k by 2.

$k = -.01755$; hence $\delta y(0) = .00468$, $\delta y(1) = 0$. Now $y(0) = -.86342$, $x(1) = .54632$, $y(1) = -.06249$, $x(2) = .04851$. $J = .64486$ so this solution is accepted as the best so far. Since we twice divided k by 2, we have sought an improvement of $.005/4 = .00125$ and obtained .00055. If we performed another iteration, we would start with $\delta J = -.00125$ and proceed from there.

7.10. $x(1) = 3$, $x(2) = 5$, $x(3) = 7$, $J = 19$.

$$\partial T_3/\partial x(3) = 1, \qquad \partial T_2/\partial x(2) = 2, \qquad \partial T_1/\partial x(1) = 3.$$
$$\partial T_2/\partial y(2) = 2, \qquad \partial T_1/\partial y(1) = 3, \qquad \partial T_0/\partial y(0) = 4.$$
$$\delta J = 4\delta y(0) + 3\delta y(1) + 2\delta y(2).$$

For $y(0) = 1 + \delta y(0)$, $y(1) = 1 + \delta y(1)$, $y(2) = 1 + \delta y(2)$, we get

$x(1) = 3 + \delta y(0)$, $\qquad x(2) = 5 + \delta y(0) + \delta y(1)$, $\qquad x(3) = 7 + \delta y(0) + \delta y(1) + \delta y(2)$.

Hence $J = 19 + 4\delta y(0) + 3\delta y(1) + 2\delta y(2)$.

Chapter 8

8.1. Let

$S(k, w) =$ the maximum value of any cargo consisting of items of types k,
$\qquad k + 1, \ldots, N$ that can be loaded on a vehicle of capacity w.

Then

$$S(k, w) = \max_{x_k = 0, 1, \ldots, [w/w_k]} [x_k v_k + S(k + 1, w - x_k w_k)],$$

where $[w/w_k]$ denotes the greatest integer less than or equal to w/w_k.

$$S(N + 1, w) = 0 \qquad \text{for} \quad w > 0.$$

8.2.

$f(3) = 7$, $\quad p(3) = 1$; $\qquad f(4) = 7$, $\quad p(4) = 1$; $\qquad f(5) = 15$, $\quad p(5) = 4$;

$f(6) = \max[7 + 7, 16 + 0, 19 - \infty, 15 + 0] = 16$, $\quad p(6) = 2$;

$f(7) = \max[7 + 7, 16 + 0, 19 + 0, 15 + 0] = 19$, $\quad p(7) = 3$;

$f(8) = \max[7 + 15, 16 + 0, 19 + 0, 15 + 7] = 22$, $\quad p(8) = 1$ or 4;

$f(9) = \max[7 + 16, 16 + 7, 19 + 0, 15 + 7] = 23$, $\quad p(9) = 1$ or 2;

$f(10) = \max[7 + 19, 16 + 7, 19 + 7, 15 + 15] = 30$, $\quad p(10) = 4$;

$f(11) = 31$, $\quad p(11) = 2$ or 4; $\qquad f(12) = 34$, $\quad p(12) = 3$ or 4;

(equations continue)

$f(13) = 37$, $p(13) = 1$ or 4; $f(14) = 38$, $p(14) = 1, 2, 3$, or 4;
$f(15) = 45$, $p(15) = 4$; $f(16) = 46$, $p(16) = 2$ or 4;
$f(17) = 49$, $p(17) = 3$ or 4; $f(18) = 52$, $p(18) = 1$ or 4;
$f(19) = 53$, $p(19) = 1, 2, 3$, or 4; $f(20) = 60$, $p(20) = 4$;
$f(21) = 61$, $p(21) = 2$ or 4; $f(22) = 64$, $p(22) = 3$ or 4.

Using the p table, we deduce the optimal cargo by making an optimal first decision, say item-type i, and then examining $p(w - w_i)$ to obtain the next decision, etc.

$w = 3$, $x_1 = 1$; $w = 4$, $x_1 = 1$; $w = 5$, $x_4 = 1$; $w = 6$, $x_2 = 1$; $w = 7$, $x_3 = 1$;
   $w = 8$, $x_1 = 1, x_4 = 1$; $w = 9$, $x_1 = 1, x_2 = 1$; $w = 10$, $x_4 = 2$;
$w = 11$, $x_2 = 1, x_4 = 1$; $w = 12$, $x_3 = 1, x_4 = 1$; $w = 13$, $x_1 = 1, x_4 = 2$;
   $w = 14$,   either $x_1 = 1, x_2 = 1, x_4 = 1$ or $x_3 = 2$; $w = 15$, $x_4 = 3$;
$w = 16$, $x_2 = 1, x_4 = 2$; $w = 17$, $x_3 = 1, x_4 = 2$; $w = 18$, $x_1 = 1, x_4 = 3$;
   $w = 19$,   either $x_1 = 1, x_2 = 1, x_4 = 2$ or $x_3 = 2, x_4 = 1$;
   $w = 20$, $x_4 = 4$; $w = 21$, $x_2 = 1, x_4 = 3$; $w = 22$, $x_3 = 1, x_4 = 3$.

8.3. The equation of the dash–dot line is $v = (v_N/w_N)(w - w_N)$. The equation of the dashed line is $v = (v_{N-1}/w_{N-1})w$. Hence

$$w^* = \frac{v_N}{(v_N/w_N) - (v_{N-1}/w_{N-1})} \;.$$

$$w' = \left[\frac{15/(3 - 2\tfrac{4}{7})}{5} \right] \cdot 5 = \left[\frac{52\tfrac{1}{2}}{5} \right] \cdot 5 = 50,$$

where $[x]$ indicates the greatest integer $\leqslant x$.

8.4. For $w = 99$, 17 items of type 4 give $w = 14$, for which $x_1 = 1$, $x_2 = 1$, $x_4 = 1$ or $x_3 = 2$, so the optimal solutions are either $x_1 = 1$, $x_2 = 1$, $x_4 = 18$ or $x_3 = 2$, $x_4 = 17$. For $w = 100$, 17 items of type 4 leave $w = 15$, for which $x_4 = 3$ is optimal, so $x_4 = 20$ is the optimal solution. Since 100 is an integer multiple of w_N (which equals 5), we could have deduced this solution with no computation of $f(w)$.

8.5.

$f(3) = 7$, $q(3) = 1$; $f(4) = 7$, $q(4) = 1$; $f(5) = 15$, $q(5) = 4$;
$f(6) = 16$, $q(6) = 2$; $f(7) = 19$, $q(7) = 3$; $f(8) = 22$, $q(8) = 4$;
$f(9) = 23$, $q(9) = 2$; $f(10) = 30$, $q(10) = 4$; $f(11) = 31$, $q(11) = 4$;
$f(12) = 34$, $q(12) = 4$; $f(13) = 37$, $q(13) = 4$; $f(14) = 38$, $q(14) = 3$ or 4;
$f(15) = 45$, $q(15) = 4$; $f(16) = 46$, $q(16) = 4$; $f(17) = 49$, $q(17) = 4$;
$f(18) = 52$, $q(18) = 4$; $f(19) = 53$, $q(19) = 4$; $f(20) = 60$, $q(20) = 4$;
                    $f(21) = 61$, $q(21) = 4$.

To compute $f(14)$, for example, we first consider item-type 1, but since $q(14 - 3)$ is 4 we do not evalutate it. Since $q(14 - 6)$ is 4, we ignore item-type 2. Since $q(7) = 3$, we evaluate item-type 3 obtaining $19 + f(7)$. Since $q(9) = 2$, we evaluate item-type 4 (we *always* evaluate item-type 4), obtaining $15 + f(9)$.

The optimal cargo is the same as in Problem 8.2.

8.6. The optimal value and policy is permanent for weights less than or equal to w, so if these q's equal N, they are correct and final. If the tentative optimal policy is N for $w + 1, w + 2, \ldots, w - z + \bar{w} - 1$, these policies will not change based on future calculations. This is because only item-type N is considered at weight w, so the policy for a larger weight

may be changed to N, but not *from* N. Similarly for weight $w + 1$, ... so we can be sure that $\bar{w}$ consecutive optimal decisions will remain N if the condition given in step 3 holds.

8.7.   For $w = 0$, steps 1 and 2 give $f(0) = 0$, $q(0) = \{0\}$ and $f^0(3) = 7$ and $q^0(3) = \{1\}$; $f^0(6) = 16$ and $q^0(6) = \{2\}$; $f^0(7) = 19$ and $q^0(7) = \{3\}$; $f^0(5) = 15$ and $q^0(5) = \{4\}$. Next we have $f(1) = 0$, $q(1) = \{0\}$ and $f^0(4) = 7$, $q^0(4) = \{1\}$; $f^0(7) = \max[16, 19] = 19$, $q^0(7)$ remains unchanged; $f^0(8) = 19$, $q^0(8) = \{3\}$; $f^0(6) = \max[15, 16] = 16$, $q^0(6)$ remains unchanged. Then $f(2) = 0$, $q(2) = \{0\}$ yields $f^0(5) = \max[7, 15] = 15$, $q^0(5)$ remains unchanged; $f^0(8) = \max[16, 19] = 19$, $q^0(8)$ remains   unchanged; $f^0(9) = 19$, $q^0(9) = \{3\}$; $f^0(7) = \max[15, 19] = 19$, $q^0(7)$ remains unchanged. Now, since $f^0(3) = 7$, we have $f(3) = 7$, $q(3) = \{1\}$ (these are now permanent, optimal values) and using $f(3)$ we get $f^0(6) = \max[14, 16] = 16$, $q^0(6)$ remains unchanged; $f^0(9) = \max[23, 19] = 23$, $q^0(9) = \{2\}$; $f^0(10) = 26$, $q^0(10) = \{3\}$; $f^0(8) = \max[22, 19] = 22$, $q^0(8) = \{4\}$. Next, $f(4) = 7$, $q(4) = \{1\}$, which yields only one new result, $f^0(11) = 26$, $q^0(11) = \{3\}$. Now $f(5) = 15$, $q(5) = \{4\}$; so, since $q(5) = \{4\}$, we only compute $f^0(10) = \max[30, 26] = 30$, $q^0(10) = \{4\}$. Now $f(6) = 16$, $q^0(6) = \{2\}$, so we use $f(6)$ to generate possible new temporary labels for $i = 2, 3,$ and 4, i.e., $w = 12, 13,$ and 11, etc.

8.8.   First $f(0) = 0$, $q(0) = \{0\}$ and $f^0(3) = 7$, $q^0(3) = \{1\}$; $f^0(6) = 16$, $q^0(6) = \{2\}$; $f^0(7) = 19$, $q^0(7) = \{3\}$; $f^0(5) = 15$, $q^0(5) = \{4\}$. Next we bring forward the previous permanent result, obtaining $f(1) = 0$, $q(1) = \{0\}$, and proceed to $w = 2$. Again we bring forward the previous result, getting $f(2) = 0$, $q(2) = \{0\}$. Next $f(3) = 7$, $q(3) = \{1\}$, and step 2 gives $f^0(6) = \max[14, 16] = 16$, $q^0(6)$ remains   unchanged; $f^0(9) = 23$, $q^0(9) = \{2\}$; $f^0(10) = 26$, $q^0(10) = \{3\}$; $f^0(8) = 22$, $q^0(8) = \{4\}$. Now we bring forward the previous result since $f^0(4) = -\infty$ and $f(3) = 7$, to obtain $f(4) = 7$, $q(4) = \{1\}$ and proceed to $w = 5$. Since $f^0(5)$ exceeds $f(4)$, we have $f(5) = 15$, $q(5) = \{4\}$; and since $q(5)$ is $\{4\}$, we compute only $f^0(10) = 30$, $q^0(10) = \{4\}$.

8.9.   Let $n(i)$, $i = 1, \ldots, 4$, represent the current "next breakpoint based on item-type i." Start with zeroth breakpoint $w = 0$, $f(0) = 0$ and $n(1) = 3.001$, $n(2) = 6.001$, $n(3) = 7.001$, $n(4) = 5.001$. Now, since $n(1)$ is the smallest $n(i)$, we consider $w = 3.001$. Taking one item of type 1, gives value $7 + f(0) = 7$ which is bigger than 0, the value of the zeroth breakpoint. Hence we record $f(3.001) = 7$ as the first breakpoint and $q(3.001) = 1$. Now we increase $n(1)$ to 6.002 which is 3.001 (the weight of item-type 1) added to 3.001 (the first breakpoint). Now $n(4)$ is the smallest $n(i)$ so 5.001 is the next potential breakpoint. Since $15 + f(0)$ is greater than 7 (the value of f at the most recently determined breakpoint), $w = 5.001$ is the second breakpoint, $f(5.001) = 15$, $q(5.001) = 4$. Now increase $n(4)$ to 8.002 which is 5.001 (the weight of item-type 4) added to 3.001 (the first breakpoint). Now 6.001, $n(2)$, is the smallest $n(i)$ and the third breakpoint is at $w = 6.001$, $f(6.001) = 16$, $q(6.001) = 2$. $n(2)$ now becomes 3.001 + 6.001 = 9.002. The current n table is $n(1) = 6.002$, $n(2) = 9.002$, $n(3) = 7.001$, $n(4) = 8.002$. $n(1)$ is the smallest, but $7 + f(3.001)$ is 14 which is less than 16, the value of the third breakpoint, so no new breakpoint is generated. We consider increasing $n(1)$ to 3.001 (the weight of item-type 1) plus 5.001 (the second breakpoint) but $q(5.001)$ is 4, which exceeds 1, so we reject this case and move on. We next consider increasing $n(1)$ to 3.001 + 6.001 (the third breakpoint), but $q(6.001) = 2$ which exceeds 1, so we again reject. Since no further breakpoints have been determined, we drop item-type 1 and $n(1)$ from all further consideration. Since the minimal $n(i)$, $i = 2, 3, 4$ is now 7.001, we examine it next, just as above, etc.

The calculation yields the breakpoint weights, optimal values and q policies shown below. It stops when items 1, 2, and 3 have been dropped from further consideration, leaving only item 4.

Breakpoint number	w	$f(w)$	$q(w)$	Breakpoint number	w	$f(w)$	$q(w)$
0	0	0	0	15	16.003	46	4
1	3.001	7	1	16	17.003	49	4
2	5.001	15	4	17	18.003	50	4
3	6.001	16	2	18	18.004	52	4
4	7.001	19	3	19	19.003	53	4
5	8.002	22	4	20	20.003	54	3
6	9.002	23	2	21	20.004	60	4
7	10.002	30	4	22	21.004	61	4
8	11.002	31	4	23	22.004	64	4
9	12.002	34	4	24	23.004	65	4
10	13.002	35	3	25	23.005	67	4
11	13.003	37	4	26	24.004	68	4
12	14.002	38	3	27	25.004	69	4
13	14.003	38	4	28	25.005	75	4
14	15.003	45	4	29	26.005	76	4

8.10. Define $S(w, u) = $ the maximum value of any cargo with weight less than or equal to w and volume less than or equal to u.

$$S(w, u) = \max_{i=1,\ldots,N} [v_i + S(w - w_i, u - u_i)].$$

$$S(w, u) = -\infty \quad \text{for} \quad w \quad \text{or} \quad u < 0.$$

Let $q(w, u) = $ the first item taken in an optimal nonincreasing enumeration. Consider only those i in the maximization for which $i \geqslant \min q(w - w_i, u - u_i)$.

Evaluate S for w's and u's chosen such that S on the right-hand side is always available when needed; for example, one might let $u = 0$ and do $w = 0, 1, \ldots, W$, then let $u = 1$ and $w = 0, 1, \ldots, W$, then $u = 2$, etc. until $u = U$ (the given available volume) and $w = W$.

The answer is $S(W, U)$.

Unless one item type is best on both value-to-weight and value-to-volume ratios, there is no way of stopping before $S(W, U)$ is calculated.

8.11. Solve as a special case of Problem 8.1. Define $S(k, w) = $ the maximum value of any solution of weight less than or equal to w consisting only of 0 or 1 items of types k, $k + 1, \ldots, N$.

$$S(k, w) = \max_{x_k = 0, 1} [x_k v_k + S(k + 1, w - x_k w_k)].$$

$$S(N + 1, w) = 0, \quad w \geqslant 0.$$

$$S(k, w) = -\infty, \quad w < 0 \quad \text{for all } k.$$

8.12. Define $S(k, \bar{w}, \bar{\bar{w}}) = $ the maximum value of any solution with the weight in knapsack $1 \leqslant \bar{w}$ and the weight in knapsack $2 \leqslant \bar{\bar{w}}$, consisting of 0 or 1 items of types $k, k + 1, \ldots, N$.

$$S(k, \bar{w}, \bar{\bar{w}}) = \max \begin{bmatrix} S(k + 1, \bar{w}, \bar{\bar{w}}) \\ v_k + S(k + 1, \bar{w} - w_k, \bar{\bar{w}}) \\ v_k + S(k + 1, \bar{w}, \bar{\bar{w}} - w_k) \end{bmatrix}.$$

$$S(N + 1, \bar{w}, \bar{\bar{w}}) = 0 \quad \text{for} \quad \bar{w} \geqslant 0 \quad \text{and} \quad \bar{\bar{w}} \geqslant 0.$$

$$S(k, \bar{w}, \bar{\bar{w}}) = -\infty \quad \text{for} \quad \bar{w} \quad \text{or} \quad \bar{\bar{w}} < 0 \quad \text{for all} \quad k.$$

8.13. The proposed procedure is incorrect. Let $N = 5$ and $w_1 = 3$, $v_1 = 100$; $w_2 = 2$, $v_2 = 8$;

$w_3 = 1$, $v_3 = 5$; $w_4 = 3$, $v_4 = 10$; $w_5 = 2$, $v_5 = 6$. The proposed procedure yields

w	$f(w)$	policy	w	$f(w)$	policy
0	0	0	6	113	1, 2, 3
1	5	3	7	115	1, 3, 4
2	8	2	8	119	1, 2, 3, 5
3	100	1	9	123	1, 2, 3, 4
4	105	1, 3	10	123	1, 2, 3, 4
5	108	1, 2			

Let $p(k, w)$ denote the number of item-type k in the optimal solution for available weight w and available items $k, k + 1, \ldots, N$.

The correct solution, based on the solution to Problem 8.11, gives

w	$S(5, w)$	$p(5, w)$	$S(4, w)$	$p(4, w)$	$S(3, w)$	$p(3, w)$	$S(2, w)$	$p(2, w)$	$S(1, w)$	$p(1, w)$
0	0	0	0	0	0	0	0	0		
1	0	0	0	0	5	1	5	0		
2	6	1	6	0	6	0	8	1		
3	6	1	10	1	11	1	13	1		
4	6	1	10	1	15	1	15	0		
5	6	1	16	1	16	0	19	1		
6	6	1	16	1	21	1	23	1		
7	6	1	16	1	21	1	24	1		
8	6	1	16	1	21	1	29	1		
9	6	1	16	1	21	1	29	1		
10	6	1	16	1	21	1	29	1	124	1

The optimal solution is $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$.

The reason for the failure of the proposed procedure is analogous to the reason that direction information had to be *part of the state description* for the turn-penalty problem of Section 7 of Chapter 1. It is also analogous to the failure of the backward procedure using only two state variables for Problem 2.12.

Often additional items of a given type have diminishing values. Any algorithm of the type suggested in this problem is then incorrect, and the problem must be solved as a general allocation problem by the procedure of Chapter 3. For an example, see Problem 10.3.

8.14. Let $f(w)$ be the optimal value function for the N-item-type problem. Let $g(w)$ be the optimal value function for the problem with $N + 1$ item types. Then

$$g(w) = \max_{x_{N+1} = 0, 1, \ldots, p} [x_{N+1}v_{N+1} + f(w - x_{N+1}w_{N+1})],$$

where p is the minimum of $[w/w_{N+1}]$ and z.

Chapter 9

9.1. Each application of (9.1) requires four additions (since the same sum appears twice), four multiplications, and a comparison. There are $N + (N - 1) + \cdots + 1$ vertices for an N-stage problem. Hence $2N(N + 1)$ additions, $2N(N + 1)$ multiplications, and $N(N + 1)/2$ comparisons are required for solution. Counting each as an operation, roughly $\frac{9}{2}N^2$ operations are required.

The deterministic problem requires roughly $\frac{3}{2} N^2$ operations, so the stochastic solution requires roughly three times the computation, not a very significant difference.

9.2. There is no correct forward procedure for a problem with stochastic transitions. One cannot go with certainty to any specific vertex, so one cannot define an expected cost of going to a certain vertex. The necessity of using a backward recursion on stochastic problems is the reason, all other considerations being equal, that we prefer to emphasize backward procedures for deterministic problems.

9.3. Define $S(x, y, z) =$ the maximum probability that the total cost is less than or equal to Z given that you start at vertex (x, y) with the cost from A to (x, y) having been z.

$S(x, y, z)$

$$= \max \begin{bmatrix} U: & p_u S(x + 1, y + 1, z + a_u(x, y)) + (1 - p_u)S(x + 1, y - 1, z + a_d(x, y)) \\ D: & (1 - p_d)S(x + 1, y + 1, z + a_u(x, y)) + p_d S(x + 1, y - 1, z + a_d(x, y)) \end{bmatrix}.$$

$$S(N, y, z) = \begin{cases} 1, & z = 0, 1, \ldots, Z, \\ 0, & \text{otherwise.} \end{cases}$$

The answer is $S(0, 0, 0)$.

9.4. We show in Figure S9.1 the optimal value function in circles at the vertices, with $G(x, y, B)$ above the vertex and $G(x, y, C)$ below. The optimal decision is shown by an arrow.

9.5. Let $F(x, y) =$ the minimum expected cost from vertex (x, y), $0 \leqslant x \leqslant x_1 - 1$, to termination. Let $G(x, y, z) =$ the minimum cost from (x, y), $x_1 \leqslant x \leqslant x_3$, to termination

Figure S9.1

given termination is known to occur at z, $z = x_3$ or x_4. Let $H(x, y) =$ the minimum expected cost from (x, y), $x_1 \leq x \leq x_2 - 1$, to termination given we do not yet know the termination stage. Let $I(x, y) =$ the minimum cost from (x, y), $x_3 \leq x \leq x_4$, to termination at x_4.

$$I(x, y) = \min \begin{bmatrix} a_u(x, y) + I(x + 1, y + 1) \\ a_d(x, y) + I(x + 1, y - 1) \end{bmatrix} \quad (x_3 \leq x \leq x_4 - 1),$$

$$I(x_4, y) = 0.$$

$$G(x, y, x_3) = \min \begin{bmatrix} a_u(x, y) + G(x + 1, y + 1, x_3) \\ a_d(x, y) + G(x + 1, y - 1, x_3) \end{bmatrix} \quad (x_1 \leq x \leq x_3 - 1),$$

$$G(x_3, y, x_3) = 0.$$

$$G(x, y, x_4) = \min \begin{bmatrix} a_u(x, y) + G(x + 1, y + 1, x_4) \\ a_d(x, y) + G(x + 1, y - 1, x_4) \end{bmatrix} \quad (x_1 \leq x \leq x_3 - 1),$$

$$G(x_3, y, x_4) = I(x_3, y).$$

$$H(x, y) = \min \begin{bmatrix} a_u(x, y) + H(x + 1, y + 1) \\ a_d(x, y) + H(x + 1, y - 1) \end{bmatrix} \quad (x_1 \leq x \leq x_2 - 2),$$

$$H(x_2 - 1, y) = \min \begin{bmatrix} a_u(x_2 - 1, y) + p_3 G(x_2, y + 1, x_3) + p_4 G(x_2, y + 1, x_4) \\ a_d(x_2 - 1, y) + p_3 G(x_2, y - 1, x_3) + p_4 G(x_2, y - 1, x_4) \end{bmatrix}.$$

$$F(x, y) = \min \begin{bmatrix} a_u(x, y) + F(x + 1, y + 1) \\ a_d(x, y) + F(x + 1, y - 1) \end{bmatrix} \quad (0 \leq x \leq x_1 - 2),$$

$$F(x_1 - 1, y)$$
$$= \min \begin{bmatrix} a_u(x_1 - 1, y) + p_1 p_3 G(x_1, y + 1, x_3) + p_1 p_4 G(x_1, y + 1, x_4) + p_2 H(x_1, y + 1) \\ a_d(x_1 - 1, y) + p_1 p_3 G(x_1, y - 1, x_3) + p_1 p_4 G(x_1, y - 1, x_4) + p_2 H(x_1, y - 1) \end{bmatrix}.$$

9.6. Define q_z $(z = 1, \ldots, N) =$ the probability that the process ends at stage z given it has not ended at stage $z - 1$. Then

$$q_z = p_z / \sum_{i=z}^{N} p_i.$$

Let $F(x, y) =$ the minimum expected cost to termination of a process starting at (x, y), $0 \leq x \leq N - 1$, and not terminated at stage x or previously.

$$F(x, y) = \min \begin{bmatrix} a_u(x, y) + (1 - q_{x+1})F(x + 1, y + 1) \\ a_d(x, y) + (1 - q_{x+1})F(x + 1, y - 1) \end{bmatrix} \quad (0 \leq x \leq N - 2),$$

$$F(N - 1, y) = \min \begin{bmatrix} a_u(x, y) \\ a_d(x, y) \end{bmatrix}.$$

9.7. The optimal decision is shown in parentheses.

$$S(2, 2, U, U) = \tfrac{9}{16}(11) + \tfrac{3}{16}(9 + 10) + \tfrac{1}{16}(13) = \tfrac{169}{16},$$

$$S(2, 2, D, U) = \tfrac{9}{16}(9) + \tfrac{3}{16}(11 + 13) + \tfrac{1}{16}(10) = \tfrac{163}{16},$$

$$S(2, 2, U, D) = \tfrac{9}{16}(10) + \tfrac{3}{16}(11 + 13) + \tfrac{1}{16}(9) = \tfrac{171}{16},$$

$$S(2, 2, D, D) = \tfrac{9}{16}(13) + \tfrac{3}{16}(9 + 10) + \tfrac{1}{16}(11) = \tfrac{185}{16}.$$

$$S(2, 0, U, U) = \tfrac{9}{16}(5) + \tfrac{3}{16}(8 + 8) + \tfrac{1}{16}(4) = \tfrac{97}{16},$$

(equations continue)

$$S(2, 0, D, U) = \tfrac{9}{16}(8) + \tfrac{3}{16}(5+4) + \tfrac{1}{16}(8) = \tfrac{107}{16},$$

$$S(2, 0, U, D) = \tfrac{9}{16}(8) + \tfrac{3}{16}(5+4) + \tfrac{1}{16}(8) = \tfrac{107}{16},$$

$$S(2, 0, D, D) = \tfrac{9}{16}(4) + \tfrac{3}{16}(8+8) + \tfrac{1}{16}(5) = \tfrac{82}{16}.$$

$$S(2, -2, U, U) = \tfrac{9}{16}(12) + \tfrac{3}{16}(8+13) + \tfrac{1}{16}(6) = \tfrac{177}{16},$$

$$S(2, -2, D, U) = \tfrac{9}{16}(8) + \tfrac{3}{16}(12+6) + \tfrac{1}{16}(13) = \tfrac{139}{16},$$

$$S(2, -2, U, D) = \tfrac{9}{16}(13) + \tfrac{3}{16}(12+6) + \tfrac{1}{16}(8) = \tfrac{179}{16},$$

$$S(2, -2, D, D) = \tfrac{9}{16}(6) + \tfrac{3}{16}(8+13) + \tfrac{1}{16}(12) = \tfrac{129}{16}.$$

$$S(1, 1, U, \_) = 3 + \min\left(\tfrac{1}{2}\cdot\tfrac{169}{16} + \tfrac{1}{2}\cdot\tfrac{97}{16}, \ \tfrac{1}{2}\cdot\tfrac{163}{16} + \tfrac{1}{2}\cdot\tfrac{107}{16}\right) = \tfrac{181}{16} \quad (U),$$

$$S(1, 1, D, \_) = 3 + \min\left(\tfrac{1}{2}\cdot\tfrac{171}{16} + \tfrac{1}{2}\cdot\tfrac{107}{16}, \ \tfrac{1}{2}\cdot\tfrac{185}{16} + \tfrac{1}{2}\cdot\tfrac{89}{16}\right) = \tfrac{185}{16} \quad (D).$$

$$S(1, -1, U, \_) = 6.5 + \min\left(\tfrac{1}{2}\cdot\tfrac{97}{16} + \tfrac{1}{2}\cdot\tfrac{177}{16}, \ \tfrac{1}{2}\cdot\tfrac{107}{16} + \tfrac{1}{2}\cdot\tfrac{139}{16}\right) = \tfrac{227}{16} \quad (D),$$

$$S(1, -1, D, \_) = 6.5 + \min\left(\tfrac{1}{2}\cdot\tfrac{107}{16} + \tfrac{1}{2}\cdot\tfrac{179}{16}, \ \tfrac{1}{2}\cdot\tfrac{89}{16} + \tfrac{1}{2}\cdot\tfrac{129}{16}\right) = \tfrac{213}{16} \quad (D).$$

$$S(0, 0, \_, \_) = 2.5 + \min\left(\tfrac{1}{2}\cdot\tfrac{181}{16} + \tfrac{1}{2}\cdot\tfrac{227}{16}, \ \tfrac{1}{2}\cdot\tfrac{185}{16} + \tfrac{1}{2}\cdot\tfrac{213}{16}\right) = \tfrac{239}{16} \quad (D).$$

The optimal decision at $(0, 0)$ is D, and whether the first transition is up or down, the optimal second decision is D.

9.8. Let $F(i, j, z) =$ the minimum expected cost from (i, j) to termination where $z = 0$ means we arrived at (i, j) from $(i - 1, j - 1)$ and $z = 1$ means we arrived at (i, j) from $(i - 1, j + 1)$.

$$F(i, j, 0) = \min\left[\begin{array}{l} U: \ \tfrac{2}{3}\{a_u(i, j) + F(i + 1, j + 1, 0)\} + \tfrac{1}{3}\{a_d(i, j) + 1 + F(i + 1, j - 1, 1)\} \\ D: \ \tfrac{1}{3}\{a_u(i, j) + F(i + 1, j + 1, 0)\} + \tfrac{2}{3}\{a_d(i, j) + 1 + F(i + 1, j - 1, 1)\} \end{array}\right].$$

$$F(i, j, 1) = \min\left[\begin{array}{l} U: \ \tfrac{2}{3}\{a_u(i, j) + 1 + F(i + 1, j + 1, 0)\} + \tfrac{1}{3}\{a_d(i, j) + F(i + 1, j - 1, 1)\} \\ D: \ \tfrac{1}{3}\{a_u(i, j) + 1 + F(i + 1, j + 1, 0)\} + \tfrac{2}{3}\{a_d(i, j) + F(i + 1, j - 1, 1)\} \end{array}\right].$$

$$F(2, 2, 0) = \min\left(\tfrac{2}{3}\cdot 3 + \tfrac{1}{3}\cdot 6, \ \tfrac{1}{3}\cdot 3 + \tfrac{2}{3}\cdot 6\right) = 4 \quad (U),$$

$$F(2, 0, 0) = \min\left(\tfrac{2}{3}\cdot 2 + \tfrac{1}{3}\cdot 4, \ \tfrac{1}{3}\cdot 2 + \tfrac{2}{3}\cdot 4\right) = \tfrac{8}{3} \quad (U),$$

$$F(2, 0, 1) = \min\left(\tfrac{2}{3}\cdot 3 + \tfrac{1}{3}\cdot 3, \ \tfrac{1}{3}\cdot 3 + \tfrac{2}{3}\cdot 3\right) = 3 \quad (U \text{ or } D),$$

$$F(2, -2, 1) = \min\left(\tfrac{2}{3}\cdot 7 + \tfrac{1}{3}\cdot 4, \ \tfrac{1}{3}\cdot 7 + \tfrac{2}{3}\cdot 4\right) = 5 \quad (D),$$

$$F(1, 1, 0) = \min\left(\tfrac{2}{3}\cdot 5 + \tfrac{1}{3}\cdot 8, \ \tfrac{1}{3}\cdot 5 + \tfrac{2}{3}\cdot 8\right) = 6 \quad (U),$$

$$F(1, -1, 1) = \min\left(\tfrac{2}{3}\cdot 5 + \tfrac{1}{3}\cdot 8, \ \tfrac{1}{3}\cdot 5 + \tfrac{2}{3}\cdot 8\right) = 6(U),$$

$$F(0, 0, \_) = \min\left(\tfrac{2}{3}\cdot 7 + \tfrac{1}{3}\cdot 6, \ \tfrac{1}{3}\cdot 7 + \tfrac{2}{3}\cdot 6\right) = \tfrac{19}{3} \quad (D).$$

9.9. $F(i, j) =$ the minimum expected cost from (i, j) to termination.

$$F(i, j) = \min\left[\begin{array}{l} U: \ \tfrac{1}{2}\{a_u(i, j) + F(i + 1, j + 1)\} + \tfrac{1}{2}\{1 + F(i, j)\} \\ D: \ \tfrac{1}{3}\{a_d(i, j) + F(i + 1, j - 1)\} + \tfrac{2}{3}\{1 + F(i, j)\} \end{array}\right].$$

$$F(1, 1) = \min\left[\begin{array}{l} \tfrac{1}{2}\cdot 4 + \tfrac{1}{2}\{1 + F(1, 1)\} \\ \tfrac{1}{3}\cdot 3 + \tfrac{2}{3}\{1 + F(1, 1)\} \end{array}\right].$$

Solving the equation $F(1, 1) = 2.5 + \tfrac{1}{2}F(1, 1)$ gives $F(1, 1) = 5$. Solving the equation

$F(1, 1) = \frac{5}{3} + \frac{2}{3}F(1, 1)$ gives $F(1, 1) = 5$. Hence at $(1, 1)$, either decision is optimal and $F(1, 1) = 5$.

$$F(1, -1) = \min\left[\begin{array}{l} \frac{1}{2}\cdot 5 + \frac{1}{2}\{1 + F(1, -1)\} \\ \frac{1}{3}\cdot 4 + \frac{2}{3}\{1 + F(1, -1)\} \end{array}\right] = 6 \quad (U \text{ or } D),$$

$$F(0, 0) = \min\left[\begin{array}{l} \frac{1}{2}\{2 + 5\} + \frac{1}{2}\{1 + F(0, 0)\} \\ \frac{1}{3}\{1 + 6\} + \frac{2}{3}\{1 + F(0, 0)\} \end{array}\right] = 8 \quad (U).$$

The optimal policy is to continue making decision U at $(0, 0)$ until transition to $(1, 1)$ occurs. Then either decision is optimal until the process ends.

The same result at each vertex can be obtained by summing an infinite series.

9.10. Define S as in Section 4. The recurrence relation is the minimum of the two alternatives in (9.1) and 0, the cost of decision S.

$$S(3, 3) = \min(2, 0, 0) = 0 \quad (D \text{ or } S),$$
$$S(3, 1) = \min(-2.5, .5, 0) = -2.5 \quad (U),$$
$$S(3, -1) = \min(.25, -1.25, 0) = -1.25 \quad (D),$$
$$S(3, -3) = \min(2.25, -1.25, 0) = -1.25 \quad (D).$$

$$S(2, 2) = \min(-\tfrac{7}{8}, -\tfrac{5}{8}, 0) = -\tfrac{7}{8} \quad (U),$$
$$S(2, 0) = \min(-\tfrac{11}{16}, -\tfrac{17}{16}, 0) = -\tfrac{17}{16} \quad (D),$$
$$S(2, -2) = \min(\tfrac{1}{4}, \tfrac{3}{4}, 0) = 0 \quad (S).$$

$$S(1, 1) = 0, \quad (S),$$
$$S(1, -1) = -\tfrac{33}{64} \quad (D).$$
$$S(0, 0) = -\tfrac{35}{256} \quad (D).$$

9.11. Let $S(x, y) =$ the minimum expected cost from (x, y) to termination given that the certain results of decisions U and D have not yet been determined at (x, y).

The probability that both U and D will mean "go up" is $\frac{3}{4}\cdot\frac{1}{4} = \frac{3}{16}$. Likewise for "go down." The probability that either U means "go up" and D means "go down" or vice versa is $\frac{10}{16}$.

$$S(x, y) = \tfrac{3}{16}\{a_u(x, y) + S(x + 1, y + 1)\} + \tfrac{3}{16}\{a_d(x, y) + S(x + 1, y - 1)\}$$
$$+ \tfrac{10}{16}\min\left[\begin{array}{l} a_u(x, y) + S(x + 1, y + 1) \\ a_d(x, y) + S(x + 1, y - 1) \end{array}\right].$$

The optimal policy, in parentheses, indicates in which direction to go if the two decisions differ.

$$S(2, 2) = 0 \quad (U \text{ or } D),$$
$$S(2, 0) = 225 \quad (D),$$
$$S(2, -2) = 12 \quad (U \text{ or } D),$$
$$S(1, 1) = \tfrac{3}{16}\cdot 225 + \tfrac{10}{16}\min(0, 225) = \tfrac{675}{16} \quad (U),$$
$$S(1, -1) = \tfrac{3}{16}\cdot 225 + \tfrac{3}{16}\cdot 12 + \tfrac{10}{16}\min(225, 12) = \tfrac{831}{16} \quad (D),$$
$$S(0, 0) = \tfrac{3}{16}(10 + \tfrac{675}{16}) + \tfrac{3}{16}(\tfrac{831}{16}) + \tfrac{10}{16}\min(\tfrac{835}{16}, \tfrac{831}{16}) = \tfrac{3327}{64} \quad (D).$$

Note that this expected cost is much smaller than the expected cost of $84\frac{1}{4}$ for the solution shown in Figure 9.2. It is always the case that if decisions are allowed to follow the

generation of random events (and depend *upon* them) rather than precede them, the expected result will be better.

Chapter 10

10.1. Define $S(i, k)$ by (10.1). Define $T(i, k, l) =$ the minimum expected cost of the remaining process if we start year k with a failed machine of age i that we have been told will cost l to restore. Then, for $i = 1, 2, \ldots,$

$$
S(i, k) = \min \left[
\begin{array}{l}
B: \quad p - t(i) + \sum_{j=0}^{J} jn(0, j) + q(0) \min \left\{ \begin{array}{l} p - u(1) + S(0, k + 1) \\ c + \sum_{l=1}^{L} m(1, l)T(1, k + 1, l) \end{array} \right\} \\
\quad + (1 - q(0))S(1, k + 1) \\
K: \quad \sum_{j=0}^{J} jn(i, j) + q(i) \min \left\{ \begin{array}{l} p - u(i + 1) + S(0, k + 1) \\ c + \sum_{l=1}^{L} m(i + 1, l)T(i + 1, k + 1, l) \end{array} \right\} \\
\quad + (1 - q(i))S(i + 1, k + 1)
\end{array}
\right],
$$

$$
T(i, k, l) = \min \left[\begin{array}{ll} B: & p - u(i) + S(0, k) \\ R: & l + S(i, k) \end{array} \right].
$$

We leave to the reader the special case $i = 0$ and boundary condition. The formula for T may be substituted in the recurrence relation for S, obtaining one, more complicated, formula rather than two.

10.2. Let $F_k(i, j) =$ the minimum expected cost of producing a total of N good bottles when k good bottles have already been produced and one mold has already produced i bottles and the other has already produced j bottles, $j \leqslant i$. Let $G_k(i) =$ the minimum expected cost of producing N good bottles when k have already been produced and one mold has already produced i good bottles and the other has failed. Let $H_k =$ the minimum expected cost of producing N good bottles when k have already been produced and both molds have failed. For i and $j > 0$,

$$
F_k(i, j) = \min \left[
\begin{array}{l}
p_i p_j F_{k+2}(i + 1, j + 1) + (1 - p_i)p_j(c_2 + G_{k+1}(j + 1)) \\
\quad + p_i(1 - p_j)(c_2 + G_{k+1}(i + 1)) + (1 - p_i)(1 - p_j)(2c_2 + H_k) \\
c_1 + c_3 + c_4 + F_k(j, 0) \\
c_1 + c_3 + c_4 + F_k(i, 0) \\
2c_1 + c_3 + c_4 + F_k(0, 0)
\end{array}
\right].
$$

$$\tag{S10.1}$$

For $j > 0$,

$$
F_k(j, 0) = p_j F_{k+2}(j + 1, 1) + (1 - p_j)(c_2 + G_{k+1}(1)).
$$
$$
F_k(0, 0) = F_{k+2}(1, 1).
$$

$$\tag{S10.2}$$

For $i > 0$,

$$
G_k(i) = \min \left[\begin{array}{l} c_1 + c_3 + c_4 + F_k(i, 0) \\ 2c_1 + c_3 + c_4 + F_k(0, 0) \end{array} \right].
$$

$$\tag{S10.3}$$

$$
H_k = 2c_1 + c_3 + c_4 + F_k(0, 0).
$$

$$\tag{S10.4}$$

At stage k, compute (S10.2), then (S10.3) and (S10.4), and finally (S10.1). The boundary condition is left to the reader.

10.3. $F_k(w)$ = the minimum expected cost of shortages of parts $k, k + 1, \ldots, N$ attainable with a transport plane of weight capacity w.

$$F_k(w) = \min_{x_k = 0, 1, \ldots, \min\{d_k, \lfloor w/w_k \rfloor\}} \left\{ c_k \sum_{z = x_k + 1}^{d_k} (z - x_k)p_k(z) + F_{k+1}(w - w_k x_k) \right\}.$$

$$F_{N+1}(w) = 0 \qquad \text{for all} \quad w \geqslant 0.$$

The expected cost saved by each additional item loaded is nonlinear, so no algorithm exists of the type of those in Chapter 8. A sequence of functions of one variable must be computed, as in the allocation problem of Chapter 3. In this problem only the stage cost, and not the state transition, is stochastic.

10.4. $F_k(x, v)$ = the maximum probability of attaining a total number of delegates from the remaining primaries greater than or equal to v where x dollars are available to be allocated sequentially to primaries $k, k + 1, \ldots, N$.

$$F_k(x, v) = \max_{x_k = 0, 1, \ldots, x} [p_k(x_k)F_{k+1}(x - x_k, v - v_k) + (1 - p_k(x_k))F_{k+1}(x - x_k, v)];$$

$$F_{N+1}(x, v) = 1 \qquad \text{if} \quad v \leqslant 0,$$
$$= 0 \qquad \text{otherwise};$$
$$F_k(x, v) = 1 \qquad \text{for all} \quad v \leqslant 0.$$

The answer is $F_1(X, V)$.

10.5. Let $S(i, x)$ = the maximum probability of finishing with at least \$5 where you start bet number i with \$$x$ and are allowed four bets.

$$S(i, x) = \max_{x_i = 0, 1, \ldots, x} [.6 \, S(i + 1, 2x_i + (x - x_i)) + .4 \, S(i + 1, x - x_i)].$$

$$S(5, x) = 1 \qquad \text{for} \quad x \geqslant 5, \qquad S(5, x) = 0 \quad \text{otherwise.}$$

$$S(i, x) = 1 \qquad \text{for} \quad x \geqslant 5 \quad \text{for all } i.$$

Let $p(i, x)$ be the optimal decision.

$$S(4, 4) = \max(0, .6, .6, .6, .6) = .6, \qquad p(4, 4) = 1, 2, 3, \text{ or } 4;$$
$$S(4, 3) = \max(0, 0, .6, .6) = .6, \qquad p(4, 3) = 2 \text{ or } 3;$$
$$S(4, 2) = 0,$$
$$S(4, 1) = 0.$$

$$S(3, 4) = \max(.6, .84, .6, .6, .6) = .84, \qquad p(3, 4) = 1;$$
$$S(3, 3) = \max(.6, .36, .6, .6) = .6, \qquad p(3, 3) = 0, 2, \text{ or } 3;$$
$$S(3, 2) = .36, \qquad p(3, 2) = 1 \text{ or } 2;$$
$$S(3, 1) = 0.$$

$$S(2, 4) = \max(.84, .84, .744, .6, .6) = .84, \qquad p(2, 4) = 0 \text{ or } 1;$$
$$S(2, 3) = \max(.6, .648, .6, .6) = .648, \qquad p(2, 3) = 1;$$
$$S(2, 2) = \max(.36, .36, .504) = .504, \qquad p(2, 2) = 2;$$
$$S(2, 1) = .216, \qquad p(2, 1) = 1.$$

$$S(1, 3) = \max(.648, .7056, .6864, .6) = .7056, \qquad p(1, 3) = 1.$$

10.6. Note that the value of this game to Elise is one minus the value to Michael. Hence, for $i = 4, 5, \ldots$,

$$S_i = \max_{x_i = 1, 2, 3} (1 - S_{i - x_i}) \qquad \text{(S10.5)}$$

and $S_1 = 0$; $S_2 = 1, p_2 = 1$; $S_3 = 1, p_3 = 2$ where p_i is the optimal pickup for the moving player and we give no optimal move if all moves lose.

Recurrence relation (S10.5), which really says that you win unless the three immediately preceding S's are all ones, now gives

$S_4 = 1, \quad p_4 = 3$;

$S_5 = 0$; $\quad S_6 = 1$; $\quad S_7 = 1, \quad p_7 = 2$; $\quad S_8 = 1, \quad p_8 = 3$;

$S_9 = 0$; $\quad S_{10} = 1, \quad p_{10} = 1$; $\quad S_{11} = 1, \quad p_{11} = 2$; $\quad S_{12} = 1, \quad p_{12} = 3$; $\quad$ etc.

The pattern is obvious: The player who is to move tries to play so that the number of remaining matchsticks after his move is of the form $4i + 1$ for integer i. If the initial position is of that form, the player who moves first loses unless his opponent fails to play optimally (which is to pick up $4 - x$ matchsticks when the moving player picks up x matchsticks).

10.7. Let $S_i =$ the maximum expected value of the game to the player about to move when there are i matchsticks remaining. A win has value 1 and a loss has value 0, so the expected value equals the probability of winning. For $i \geqslant 2$,

$$S_i = \max_{x_i = 1, 2, 3} \left[\tfrac{1}{2}(1 - S_{i-x_i-1}) + \tfrac{1}{2}(1 - S_{i-x_i}) \right]$$

and $S_{-2} = 1$, $S_{-1} = 1$, $S_0 = 1$, $S_1 = 0$. Hence

$S_2 = \max(\tfrac{1}{2}, 0, 0) = \tfrac{1}{2}$, $\qquad p_2 = 1$;

$S_3 = \max(\tfrac{3}{4}, \tfrac{1}{2}, 0) = \tfrac{3}{4}$, $\qquad p_3 = 1$;

$S_4 = \max(\tfrac{3}{8}, \tfrac{3}{4}, \tfrac{1}{2}) = \tfrac{3}{4}$, $\qquad p_4 = 2$;

$S_5 = \max(\tfrac{1}{4}, \tfrac{3}{8}, \tfrac{3}{4}) = \tfrac{3}{4}$, $\qquad p_5 = 3$;

$S_6 = \max(\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{3}{8}) = \tfrac{3}{8}$, $\qquad p_6 = 3$;

$S_7 = \max(\tfrac{7}{16}, \tfrac{1}{4}, \tfrac{1}{4}) = \tfrac{7}{16}$, $\qquad p_7 = 1$; $\quad$ etc.

The pattern is not obvious, but the computation can be carried further and the optimal policy can be memorized.

10.8. Let P denote some subset of the numbers $1, 2, 3, \ldots, 9$. Let $Q(P)_k$ denote the set of all subsets of P whose elements sum to k, $k = 2, \ldots, 12$. Let $S(P) =$ Louise's minimum expected score before rolling the dice, starting with the set P of markers. Let $p(k) =$ the probability that the dice sum to k, $k = 2, \ldots, 12$. Then Louise's recurrence relation is

$$S(P) = \sum_{k=2}^{12} p(k) \min_{R \in Q(P)_k} S(P - R).$$

If $Q(P)_k$ is empty, the minimum on the right is the sum of the elements of P. $S(\varnothing) = 0$. S is computed first for all sets P with one element, then with two elements, then three, etc., until S is computed for the one set having nine elements. The minimizing set R gives the optimal policy for each set P and roll k.

For example,

$S(\{1\}) = 1$, $\qquad S(\{2\}) = \tfrac{1}{36} \cdot 0 + \tfrac{35}{36} \cdot 2 = \tfrac{35}{18}$, $\qquad S(\{3\}) = \tfrac{2}{36} \cdot 0 + \tfrac{34}{36} \cdot 3 = \tfrac{17}{6}$, $\qquad$ etc.

Then

$$S(\{1, 2\}) = \tfrac{1}{36} \cdot 1 + \tfrac{2}{36} \cdot 0 + \tfrac{33}{36} \cdot 3,$$

$$S(\{1, 3\}) = \tfrac{2}{36} \cdot 1 + \tfrac{3}{36} \cdot 0 + \tfrac{31}{36} \cdot 4,$$

$$S(\{2, 3\}) = \tfrac{1}{36} \cdot \tfrac{17}{6} + \tfrac{2}{36} \cdot \tfrac{35}{18} + \tfrac{4}{36} \cdot 0 + \tfrac{29}{36} \cdot 5, \qquad \text{etc.}$$

Next

$$S(\{1, 2, 3\}) = \tfrac{1}{36} S(\{1, 3\}) + \tfrac{2}{36} \min(S(\{1, 2\}), S(\{3\})) + \tfrac{3}{36} S(\{2\})$$
$$+ \tfrac{4}{36} S(\{1\}) + \tfrac{5}{36} \cdot 0 + \tfrac{21}{36} \cdot 6, \qquad \text{etc.}$$

10.9. Let $S(P, z) =$ the maximum probability that Will's score is less than z, Louise's score, starting with the set P of markers remaining. Will's recurrence relation is

$$S(P, z) = \sum_{k=2}^{12} p(k) \max_{R \in Q(P)_k} S(P - R, z).$$

$$S(\emptyset, z) = \begin{bmatrix} 1 & \text{if} & z > 0, \\ 0 & \text{if} & z = 0. \end{bmatrix}$$

$S(P, z) = 1$ for all P whose markers sum to less than z. For other P, if $Q(P)_k = \emptyset$, the maximum on the right is equal to 0.

For example, for z equal to, say, 3,

$$S(\{1\}, 3) = S(\{2\}, 3) = 1; \qquad S(\{3\}, 3) = \tfrac{2}{36} \cdot 1 + \tfrac{34}{36} \cdot 0 = \tfrac{1}{18}, \qquad \text{etc.}$$

$$S(\{1, 2\}, 3) = \tfrac{1}{36} \cdot 1 + \tfrac{2}{36} \cdot 1 + \tfrac{33}{36} \cdot 0,$$
$$S(\{1, 3\}, 3) = \tfrac{2}{36} \cdot 1 + \tfrac{3}{36} \cdot 1 + \tfrac{31}{36} \cdot 0,$$
$$S(\{2, 3\}, 3) = \tfrac{1}{36} \cdot \tfrac{1}{18} + \tfrac{2}{36} \cdot 1 + \tfrac{4}{36} \cdot 1 + \tfrac{29}{36} \cdot 0, \qquad \text{etc.}$$

$$S(\{1, 2, 3\}, 3) = \tfrac{1}{36} S(\{1, 3\}, 3) + \tfrac{2}{36} \max(S(\{1, 2\}, 3), S(\{3\}, 3))$$
$$+ \tfrac{3}{36} S(\{2\}, 3) + \tfrac{4}{36} S(\{1\}, 3) + \tfrac{5}{36} S(\emptyset, 3)$$
$$+ \tfrac{21}{36} \cdot 0, \qquad \text{etc.}$$

10.10. Let J be the set $\{1, \ldots, N\}$. Let P_j be a set of j elements of J, representing j men. Let $S(P_j) =$ the maximum expected value for the process starting with the set P_j uninvited and no one has yet accepted.

$$S(P_j) = \max_{i \in P_j} \left[p_{i, N-j+1} v_i + (1 - p_{i, N-j+1}) S(P_j - \{i\}) \right].$$

$$S(\emptyset) = 0.$$

Applying the formulas to the case $N = 2$, $p_{11} = \tfrac{3}{5}$, $p_{12} = \tfrac{1}{2}$, $v_1 = 10$, $p_{21} = \tfrac{1}{2}$, $p_{22} = 0$, $v_2 = 8$:

$$S(\{1\}) = 5, \qquad S(\{2\}) = 0;$$
$$S(\{1, 2\}) = \max(\tfrac{3}{5} \cdot 10 + \tfrac{2}{5} \cdot 0, \tfrac{1}{2} \cdot 8 + \tfrac{1}{2} \cdot 5) = 6\tfrac{1}{2}$$

and man 2 is the first to be asked.

The moral is: pride goeth before a party.

This problem and the previous two are similar to the traveling-salesman problem in that the state is a set of elements. Such problems are not computationally feasible for more than a small number of elements that may be members of the state set.

Chapter 11

11.1. Let

$f_i(w) =$ the expected total discounted cost for periods i through n (other than
    the cost due to the delivery of z_{i-1}) given that w items will be on
    hand in period i and an optimal ordering policy is followed.

$$f_i(w) = \min_{y > w} \left[ac_i(y - w) + L_i^0(w) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y - d) p_i(d) \right] \qquad (i = 1, 2, \ldots, n-1),$$

$$f_n(w) = L_n^0(w) + \alpha \sum_{d=0}^{\infty} t(w - d) p_n(d).$$

Assuming that an order is not outstanding at the beginning of period 1, then $w_1 = x_1$ and the answer is $f_1(x_1)$.

11.2.

$$f_3(1, 0) = \tfrac{5}{8} ; \qquad f_3(-1, 0) = 14\tfrac{1}{4} ;$$

$$f_3(1, 1) = -\tfrac{3}{4} ; \qquad f_3(-1, 1) = 7\tfrac{1}{4} ;$$

$$f_3(0, 0) = 4\tfrac{1}{4} ; \qquad f_3(-2, 0) = 24\tfrac{1}{4} ;$$

$$f_3(0, 1) = \tfrac{5}{8} ; \qquad f_3(-2, 1) = 17\tfrac{1}{4} .$$

$$f_2(0, 0) = \min[c_2(0) + L_2^0(0) + \{ \tfrac{3}{4} f_3(0, 0) + \tfrac{1}{4} f_3(-1, 0)\}, \; c_2(1) + L_2^0(0)$$

$$+ \{ \tfrac{3}{4} f_3(0, 1) + \tfrac{1}{4} f_3(-1, 1)\}]$$

$$= \min[0 + \tfrac{3}{4} + \{ \tfrac{51}{16} + \tfrac{57}{16} \}, \; 3 + \tfrac{3}{4} + \{ \tfrac{15}{32} + \tfrac{29}{16} \}]$$

$$= \min[7\tfrac{1}{2} , 6\tfrac{1}{32}] = 6\tfrac{1}{32} , \qquad z_2(0, 0) = 1;$$

$$f_2(0, 1) = 2\tfrac{9}{32} , \qquad z_2(0, 1) = 0;$$

$$f_2(-1, 0) = 16\tfrac{1}{2} , \qquad z_2(-1, 0) = 1;$$

$$f_2(-1, 1) = 9\tfrac{1}{32} , \qquad z_2(-1, 1) = 1.$$

$$f_1(0,0) = \min[c_1(0) + L_1^0(0) + \{ \tfrac{3}{4} f_2(0,0) + \tfrac{1}{4} f_2(-1,0)\}, \; c_1(1) + L_1^0(0)$$

$$+ \{ \tfrac{3}{4} f_2(0,1) + \tfrac{1}{4} f_2(-1,1)\}]$$

$$= \min[0 + \tfrac{3}{4} + \{ \tfrac{579}{128} + \tfrac{33}{8} \}, \; 3 + \tfrac{3}{4} + \{ \tfrac{219}{128} + \tfrac{289}{128} \}]$$

$$= \min[9\tfrac{51}{128} , 7\tfrac{23}{32}] = 7\tfrac{23}{32} , \qquad z_1(0,0) = 1.$$

11.3. Let $f_i(w, z_{i-m+1}, \ldots, z_{i-1}) =$ the expected total discounted cost for periods i through n given that w items are on hand in period i (after the delivery of any previous orders), z_j items were ordered in period j $(j = i - m + 1, i - m + 2, \ldots, i - 1)$ but have not yet been delivered, and an optimal ordering policy is followed.

$$f_i(w, z_{i-m+1}, \ldots, z_{i-1})$$

$$= \min_{z_i > 0} \left[c_i(z_i) + L_i^0(w) \right.$$

$$\left. + \alpha \sum_{k=1}^{m} \left\{ \sum_{d=0}^{\infty} f_{i+1}\left(w + \sum_{j=i-m+1}^{i-k+1} z_j - d, 0, \ldots, 0, z_{i-k+2}, \ldots, z_i \right) p_i(d) \right\} r(k) \right]$$

$$(i = 1, 2, \ldots, n - 1).$$

$$f_n(w, z_{n-m+1}, \ldots, z_{n-1}) = L_n^0(w) + \alpha \sum_{k=1}^{m} \left[\sum_{d=0}^{\infty} t\left(w + \sum_{j=n-m+1}^{n-k+1} z_j - d \right) p_n(d) \right] r(k),$$

where $z_n \equiv 0$. The answer is $f_1(x_1, 0, \ldots, 0)$.

11.4. Let $f_i(x, z_{i-1}) =$ the expected total discounted cost for periods i through n given that you start period i with x items on hand, z_{i-1} items were ordered in period $i - 1$, and an optimal ordering policy is followed.

$$f_i(x, z_{i-1}) = \min_{z > 0} \left[c_i(z) + L_i^0(x + z) + a_i(z_{i-1}, z) \right.$$

$$\left. + \alpha \sum_{d=0}^{\infty} f_{i+1}(x + z - d, z) p_i(d) \right] \qquad (i = 1, 2, \ldots, n),$$

$$f_{n+1}(x, z_n) = t(x).$$

The answer is $f_1(x_1, 0)$.

11.5. Let $f_i(x)$ = the expected total discounted profit for periods i through n given that you start period i with x items on hand and an optimal ordering policy is followed.

If $x > 0$, then

$$f_i(x) = \max_{y > x} \left[-c_i(y-x) - L_i^0(y) + r \sum_{d=0}^{y} dp_i(d) + ry \sum_{d=y+1}^{\infty} p_i(d) \right.$$

$$\left. + \alpha \sum_{d=0}^{\infty} f_{i+1}(y-d)p_i(d) \right] \qquad (i = 1, 2, \ldots, n).$$

If $x < 0$, then

$$f_i(x) = \max \left[\begin{array}{l} \displaystyle\max_{x < y < 0} \left\{ -c_i(y-x) - L_i^0(y) + r\cdot(y-x) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y-d)p_i(d) \right\} \\[3mm] \displaystyle\max_{y > 0} \left\{ -c_i(y-x) - L_i^0(y) + r\cdot(-x) + r \sum_{d=0}^{y} dp_i(d) \right. \\[3mm] \left. + ry \sum_{d=y+1}^{\infty} p_i(d) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y-d)p_i(d) \right\} \end{array} \right]$$

$$(i = 1, 2, \ldots, n).$$

The boundary condition is

$$f_{n+1}(x) = \left[\begin{array}{ll} s(x) & \text{if} \quad x > 0 \\ -b(-x) + r\cdot(-x) & \text{if} \quad x < 0 \end{array} \right.$$

and the answer is $f_1(x_1)$.

11.6. Let $f_i(x, u)$ = the expected total discounted cost for periods i through n given that you start period i with x items on hand, u of these items were delivered two periods ago (if $x < 0$, then $u = 0$), and an optimal ordering policy is followed.

If $x > 0$ and $x > u > 0$, then

$$f_i(x, u) = \min_{y > x} \left[c_i(y-x) + L_i^0(y) + \alpha \left\{ \sum_{d=0}^{u} f_{i+1}(y-u, x-u)p_i(d) \right. \right.$$

$$\left. \left. + \sum_{d=u+1}^{x} f_{i+1}(y-d, x-d)p_i(d) + \sum_{d=x+1}^{\infty} f_{i+1}(y-d, 0)p_i(d) \right\} \right]$$

$$(i = 1, 2, \ldots, n).$$

If $x < 0$, then

$$f_i(x, 0) = \min_{y > x} \left[c_i(y-x) + L_i^0(y) + \alpha \sum_{d=0}^{\infty} f_{i+1}(y-d, 0)p_i(d) \right] \qquad (i = 1, 2, \ldots, n).$$

The boundary condition is $f_{n+1}(x, u) = t(x)$ and the answer is $f_1(x_1, u_1)$.

11.7. Let $f_i(x)$ = the expected total discounted cost for periods i through the terminal period given that you start period i with x items on hand, you have not terminated previously, and an optimal ordering policy is followed.

Let

$$r(i, i) = r(i) \Big/ \sum_{j=i}^{n} r(j) \qquad \text{for} \quad i = 1, 2, \ldots, n.$$

$$f_i(x) = \min_{y > x} \left[c_i(y-x) + L_i^0(y) + \alpha \left\{ r(i, i) \sum_{d=0}^{\infty} t(y-d)p_i(d) \right. \right.$$

$$\left. \left. + [1 - r(i, i)] \sum_{d=0}^{\infty} f_{i+1}(y-d)p_i(d) \right\} \right] \qquad (i = 1, 2, \ldots, n-1).$$

$$f_n(x) = \min_{y > x} \left[c_n(y - x) + L_n^0(y) + \alpha \sum_{d=0}^{\infty} t(y - d)p_n(d) \right].$$

The answer is $f_1(x_1)$.

11.8. Let $f_i(x) =$ the expected total discounted cost for periods i through n given that you start period i with a cash balance of $\$x$ and an optimal policy is followed.

$$L_i^0(w) = \sum_{d = -\infty}^{w} c_h \cdot (w - d) p_i(d) + \sum_{d = w+1}^{\infty} c_p \cdot (d - w) p_i(d) \qquad \text{for all} \quad w.$$

The recurrence relation is

$$f_i(x) = \min \begin{bmatrix} \min_{y > x} \left\{ c_s \cdot (y - x) + L_i^0(y) + \alpha \sum_{d = -\infty}^{\infty} f_{i+1}(y - d)p_i(d) \right\} \\ L_i^0(x) + \alpha \sum_{d = -\infty}^{\infty} f_{i+1}(x - d)p_i(d) \\ \min_{y < x} \left\{ c_b \cdot (x - y) + L_i^0(y) + \alpha \sum_{d = -\infty}^{\infty} f_{i+1}(y - d)p_i(d) \right\} \end{bmatrix} \qquad (i = 1, 2, \ldots, n)$$

and the boundary condition is $f_{n+1}(x) = 0$. The answer is $f_1(x_1)$.

Chapter 12

12.1. It is true for $k = 2$ by the definition of a convex set. Therefore, assume it is true for k. Let $c = \sum_{i=1}^{k} \lambda^i$ and $\gamma^i = \lambda^i/c$ for $i = 1, 2, \ldots, k$.

$$x = \sum_{i=1}^{k+1} \lambda^i x^i = \sum_{i=1}^{k} \lambda^i x^i + \lambda^{k+1} x^{k+1} = c \sum_{i=1}^{k} \gamma^i x^i + \lambda^{k+1} x^{k+1} \in S.$$

12.2. It is true for $k = 2$ by the definition of a concave function. Therefore, assume it is true for k. Let $c = \sum_{i=1}^{k} \lambda^i$ and $\gamma^i = \lambda^i/c$ for $i = 1, 2, \ldots, k$.

$$f\left(\sum_{i=1}^{k+1} \lambda^i x^i \right) = f\left(c \sum_{i=1}^{k} \gamma^i x^i + \lambda^{k+1} x^{k+1} \right) \geq cf\left(\sum_{i=1}^{k} \gamma^i x^i \right) + \lambda^{k+1} f(x^{k+1})$$

$$\geq c \sum_{i=1}^{k} \gamma^i f(x^i) + \lambda^{k+1} f(x^{k+1}) = \sum_{i=1}^{k+1} \lambda^i f(x^i).$$

12.3. (a) Convex means $\alpha f(x) + (1 - \alpha)f(y) \geq f(\alpha x + (1 - \alpha)y)$, while 0-convex is equivalent to

$$\frac{b}{a + b} f(z + a) + \frac{a}{a + b} f(z - b) \geq f(z).$$

To show that convex implies 0-convex, let $\alpha = b/(a + b)$, $x = z + a$, and $y = z - b$. To show that 0-convex implies convex, let $a = (1 - \alpha)(x - y)$, $b = \alpha(x - y)$, and $z = \alpha x + (1 - \alpha)y$.

(b) Obvious.

(c) f K-convex and $\alpha > 0$ implies

$$\alpha K + \alpha f(x + a) - \alpha f(x) - \alpha(a/b)[f(x) - f(x - b)] \geq 0.$$

g M-convex and $\beta > 0$ implies

$$\beta M + \beta g(x + a) - \beta g(x) - \beta(a/b)[g(x) - g(x - b)] \geq 0.$$

Adding the two inequalities gives

$$(\alpha K + \beta M) + (\alpha f + \beta g)(x + a) - (\alpha f + \beta g)(x) - (a/b)[(\alpha f + \beta g)(x) - (\alpha f + \beta g)(x - b)] \geq 0.$$

(d)

$$K + g(x + a) - g(x) - (a/b)[g(x) - g(x - b)]$$
$$= K + f((x - z) + a) - f(x - z) - (a/b)[f(x - z) - f((x - z) - b)] \geq 0.$$

(e)
$$K + g(x + a) - g(x) - (a/b)[g(x) - g(x - b)]$$

$$= K + \int_0^\infty f(x + a - z)\phi(z)\, dz - \int_0^\infty f(x - z)\phi(z)\, dz$$

$$- \frac{a}{b}\left[\int_0^\infty f(x - z)\phi(z)\, dz - \int_0^\infty f(x - b - z)\phi(z)\, dz\right]$$

$$= \int_0^\infty \left\{ K + f(x - z + a) - f(x - z) - \frac{a}{b}[f(x - z) - f(x - z - b)]\right\}\phi(z)\, dz > 0.$$

12.4.

$$
\begin{array}{llllll}
c(1, 1) = 5, & j = 1; & c(2, 2) = 9, & j = 2; & c(3, 3) = 9, & j = 3; \\
c(1, 2) = 14, & j = 1 \text{ or } 2; & c(2, 3) = 16, & j = 2; & c(3, 4) = 13, & j = 3; \\
c(1, 3) = 20, & j = 2; & c(2, 4) = 20, & j = 2; & c(4, 4) = 5, & j = 4. \\
c(1, 4) = 24, & j = 2; & & & &
\end{array}
$$

$$f_0 = 0;$$
$$f_1 = 0 + 5 = 5, \qquad j(1) = 1;$$
$$f_2 = \min[0 + 14,\ 5 + 9] = 14, \qquad j(2) = 1 \text{ or } 2;$$
$$f_3 = \min[0 + 20,\ 5 + 16,\ 14 + 9] = 20, \qquad j(3) = 1;$$
$$f_4 = \min[0 + 24,\ 5 + 20,\ 14 + 13,\ 20 + 5] = 24, \qquad j(4) = 1.$$

Therefore, periods 0 and 4 are r.p.'s and an optimal production policy is $z_1 = 0$, $z_2 = 6$, $z_3 = 0$, $z_4 = 0$.

12.5. Let $d(j, i) = c(j, i) - c(j, i - 1)$.

$$d(j, i) = d_i c_j + d_i \sum_{l=j}^{i-2} h_l + d_i h_{i-1} = d_i\left[c_j + \sum_{l=j}^{i-1} h_l\right],$$

$$d(k, i) = d_i\left[c_k + \sum_{l=k}^{i-1} h_l\right].$$

Therefore,

$$d(j, i) - d(k, i) = d_i\left[c_j - c_k \sum_{l=j}^{k-1} h_l\right] > 0.$$

12.6. Let $h(y) \equiv \int_0^\infty f_i(y - z)\phi_{i-1}(z)\, dz$. For $y < s_i$, it is easy to show that $h(y)$ is linear and thus continuous. For $y > s_i$, $h(y)$ is given by

$$h(y) = \int_0^{y-s_i} g_i(y - z)\phi_{i-1}(z)\, dz + [g_i(S_i) + K_i]\int_{y-s_i}^\infty \phi_{i-1}(z)\, dz - c_i y + c_i E(D_{i-1}).$$

Since $g_i(y)$ is continuous, it is bounded and uniformly continuous on closed and bounded intervals. These facts can be used to show that the first integral is continuous. Furthermore, the second integral is continuous since $\phi_{i-1}(z)$ is integrable. The proof for $y = s_i$ is similar.

Chapter 13

13.1. Suppose that there exists another function V such that

$$V(i) = C[i, f(i)] + \alpha \sum_{j=0}^m P_{ij}[f(i)]V(j), \qquad i \in I.$$

Then it is easy to show by induction that $T_f^n V = V$ for all $n > 1$. Letting $n \to \infty$ and using part (iii) of Lemma 13.2 gives $V = V_{f, \alpha}$, which is the desired result.

13.2. The proof is by induction.

$$|V_\alpha(i, 0) - V_\alpha(i)| = |V_\alpha(i)| < \alpha^0 M/(1 - \alpha).$$

Assume that

$$|V_\alpha(i, n - 1) - V_\alpha(i)| < \alpha^{n-1} M/(1 - \alpha).$$

Let $\bar{a}_i$ be the action that minimizes

$$C(i, a) + \alpha \sum_{j=0}^m P_{ij}(a) V_\alpha(j)$$

and let a_i^n be the action that minimizes

$$C(i, a) + \alpha \sum_{j=0}^m P_{ij}(a) V_\alpha(j, n - 1).$$

Then

$$V_\alpha(i, n) - V_\alpha(i) < \alpha \sum_{j=0}^m P_{ij}(\bar{a}_i)[V_\alpha(j, n - 1) - V_\alpha(j)]$$

$$< \alpha \sum_{j=0}^m P_{ij}(\bar{a}_i) \frac{\alpha^{n-1} M}{1 - \alpha} = \frac{\alpha^n M}{1 - \alpha}$$

and

$$V_\alpha(i) - V_\alpha(i, n) < \alpha \sum_{j=0}^m P_{ij}(a_i^n)[V_\alpha(j) - V_\alpha(j, n - 1)]$$

$$< \alpha \sum_{j=0}^m P_{ij}(a_i^n) \frac{\alpha^{n-1} M}{1 - \alpha} = \frac{\alpha^n M}{1 - \alpha}.$$

Therefore, $|V_\alpha(i, n) - V_\alpha(i)| < \alpha^n M/(1 - \alpha)$ for all $n > 1$. Note that the same proof is valid for $i \in I$.

13.3.

$$f_1(0) = K, \qquad f_1(1) = K, \qquad f_1(2) = K, \qquad f_1(3) = R;$$

$$V_{f_1}(0) = 9503.50, \qquad V_{f_1}(1) = 10{,}754.88, \qquad V_{f_1}(2) = 12{,}068.11, \qquad V_{f_1}(3) = 12{,}503.50.$$

$$f_2(0) = K, \qquad f_2(1) = K, \qquad f_2(2) = R, \qquad f_2(3) = R;$$

$$V_{f_2}(0) = 9368.08, \qquad V_{f_2}(1) = 10{,}602.98, \qquad V_{f_2}(2) = 11{,}868.08, \qquad V_{f_2}(3) = 12{,}368.08.$$

$$f_3(0) = K, \qquad f_3(1) = K, \qquad f_3(2) = R, \qquad f_3(3) = R.$$

Therefore, the optimal policy is to keep in states 0, 1 and replace in states 2, 3.

13.4. Let state 3 and state ∞ be the same.

$$f_1(0) = A, \qquad f_1(1) = A, \qquad f_1(2) = A;$$

$$V_{f_1}(0) = -38{,}000, \qquad V_{f_1}(1) = -40{,}000, \qquad V_{f_1}(2) = -42{,}000, \qquad V_{f_1}(3) = 0.$$

$$f_2(0) = R, \qquad f_2(1) = A, \qquad f_2(2) = A;$$

$$V_{f_2}(0) = -39{,}599.01, \qquad V_{f_2}(1) = -40{,}000, \qquad V_{f_2}(2) = -42{,}000, \qquad V_{f_2}(3) = 0.$$

$$f_3(0) = R, \qquad f_3(1) = A, \qquad f_3(2) = A.$$

Therefore, the optimal policy is reject an offer of 38,000 and accept an offer of 40,000 or

42,000. Steffi's expected profit is

$$\tfrac{1}{2}(39,599.01) + \tfrac{3}{8}(40,000) + \tfrac{1}{8}(42,000) = 40,049.51.$$

13.5.

n	$V_\alpha(0, n)$	$V_\alpha(1, n)$	$V_\alpha(2, n)$	$V_\alpha(3, n)$	$g_n(0)$	$g_n(1)$	$g_n(2)$	$g_n(3)$
1	100.00	200.00	500.00	3100.00	1	1	1	2
2	236.56	615.63	1592.50	3236.56	1	1	1	2
3	472.76	1114.47	2403.34	3472.76	1	1	1	2

The optimal policy is obtained for $n \geqslant 6$ and $V_\alpha(i, n)$ differs from $V_\alpha(i)$ by less than 1% for $n \geqslant 93$.

13.6. Since K is finite, there must be an infinite number of values of n for which the action taken in state i, $g_n(i)$, is the same, say a_i. (Note that a_i may not be unique.) Let $n_1, n_2, \ldots$ denote this sequence. For any member of this sequence, we have

$$V_\alpha(i, n_k) = C(i, a_i) + \alpha \sum_{j=0}^{m} P_{ij}(a_i)V_\alpha(j, n_k - 1).$$

Letting $k \to \infty$, we get

$$V_\alpha(i) = C(i, a_i) + \alpha \sum_{j=0}^{m} P_{ij}(a_i)V_\alpha(j).$$

It now follows from Theorem 13.3 that the stationary policy that chooses action a_i when the process is in state i is α-optimal.

Let $A(i)$ be the set of all actions that occur infinitely often in the sequence $g_1(i)$, $g_2(i)$, $\ldots$. Since K is finite, it also follows that there exists an $n(i)$ such that $g_n(i) \in A(i)$ for $n > n(i)$. Let $n^* = \max_{i \in I_m} n(i)$. It follows that g_n is α-optimal for $n > n^*$.

Chapter 14

14.1. Define $V_i(x)$, for this new process, as in (14.3).

$$V_i(x) = \underset{z(i)}{E} \underset{y(x, z(i))}{\min} \left\{ a(i)x^2 + c(i)y^2 + V_{i+1}(g(i)x + h(i)y + z(i)) \right\}.$$

Note that the order of the minimum and expectation are reversed and that y is now allowed to depend on $z(i)$. Boundary condition (14.5) is unchanged.

14.2. Assume $V_{i+1}(x) = \dot{p}(i + 1)x^2 + q(i + 1)x + r(i + 1)$, where p, q, and r are not necessarily the same as in the text.

$$V_i(x) = \underset{z}{E} \underset{y}{\min} \left\{ ax^2 + cy^2 + p(gx + hy + z)^2 + q(gx + hy + z) + r \right\}.$$

$$y = -\frac{2pghx + 2phz + qh}{2(c + ph^2)}.$$

Note that y depends on z in this formula.

$$V_i(z) = \underset{z}{E} \left\{ \left(a + \frac{pcg^2}{c + ph^2} \right)x^2 + \left(\frac{2pcgz + qcg}{c + ph^2} \right)x + \frac{4cpz^2 + 4qcz - h^2q^2}{4(c + ph^2)} + r \right\}$$

$$= \left(a + \frac{pcg^2}{c + ph^2} \right)x^2 + \left(\frac{2pcg\bar{z} + qcg}{c + ph^2} \right)x + \frac{4pc\bar{z}^2 + 4pc\sigma^2 + 4qc\bar{z} - h^2q^2}{4(c + ph^2)} + r.$$

After some algebra, we find that p and q satisfy (14.19) and (14.20) but that r satisfies a

formula different from (14.21) and is smaller for this new problem.

Since y depends on z, not on $\bar{z}$, certainty equivalence does not hold.

14.3. If $p(i+1)$ is negative (but $c(i) + p(i+1)h^2(i)$ is positive so that the problem has a minimum), then randomness reduces the expected cost. Such a problem is:

$$J = x^2(0) + 4y^2(0) - x^2(1) - y^2(1) + 2x^2(2),$$

$$x(i+1) = x(i) + y(i) + z(i) \qquad (i = 0, 1),$$

$$x(0) = 0,$$

$$E(z(i)) = 0, \qquad \text{var}(z(i)) = 1 \qquad (i = 0, 1).$$

Then, for either the stochastic or deterministic problem,

$$p(2) = 2, \qquad p(1) = -1 + 2 - \frac{4}{-1+2} = -3, \qquad p(0) = 1 - 3 - \frac{9}{4-3} = -11.$$

$$q(1) = 0, \qquad q(0) = 0.$$

For the deterministic problem ($z = \bar{z} = 0$)

$$r(1) = 0, \qquad r(0) = 0.$$

For the stochastic problem

$$r(1) = 2, \qquad r(0) = -1.$$

The cost, starting at $x(0) = 0$, of the deterministic problem is zero and the expected cost of the stochastic problem is -1. If $p(i+1)$ is negative, the minimum cost function V_{i+1} is concave. Hence it lies beneath its tangent so the cost increases less if the expected next state $\bar{x}(i+1)$ is changed by ϵ in the direction of increasing V_{i+1} (δ_1 in Figure S14.1) than it decreases if $\bar{x}(i+1)$ is changed by ϵ in the direction of decreasing V_{i+1} (δ_2 in Figure S14.1). Hence the random fluctuation about the expected state $\bar{x}(i+1)$ due to the randomness of $z(i)$ reduces the expected cost.

Figure S14.1

14.4. The results are

$$y(i) = -\frac{2u(i+1)\bar{g}(i)\bar{h}(i)x + 2u(i+1)\bar{h}(i)\bar{z}(i) + v(i+1)\bar{h}(i)}{2\left[\bar{c}(i) + u(i+1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\right]},$$

$$u(i) = \bar{a}(i) + u(i+1)\left(\bar{g}^2(i) + \sigma_g^2(i)\right) - \frac{u^2(i+1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\left(\bar{g}^2(i) + \sigma_g^2(i)\right)}{\bar{c}(i) + u(i+1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)},$$

$$v(i) = 2u(i+1)\bar{g}(i)\bar{z}(i) + v(i+1)\bar{g}(i)$$

$$- \frac{2u^2(i+1)\bar{g}(i)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\bar{z}(i) + u(i+1)\bar{g}(i)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)v(i+1)}{\bar{c}(i) + u(i+1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)},$$

$$w(i) = u(i + 1)(\bar{z}^2(i) + \sigma_z^2(i)) + v(i + 1)\bar{z}(i) + w(i + 1)$$

$$- \frac{4u^2(i + 1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\bar{z}^2(i) + 4u(i + 1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\bar{z}(i)v(i + 1)}{4\left[\bar{c}(i) + u(i + 1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)\right]} \overset{+ v^2(i + 1)\left(\bar{h}^2(i) + \sigma_h^2(i)\right)}{}.$$

Certainty equivalence does not hold, due to the σ_g^2 and σ_h^2 terms. If just a, c, and l, as well as z, were random, certainty equivalence would still hold.

14.5. Let $F_i(x)$, $i = 0, \ldots, N_3 - 1$, be the minimum expected cost for a process starting stage i in state $x(i) = x$ with probability v of terminating at stage N_1 and $1 - v$ of terminating at stage N_2. Let $G_i(x)$, $i = N_3, \ldots, N_1$, be the minimum cost of a process starting stage i in state $x(i) = x$ and terminating at stage N_1. Let $H_i(x)$, $i = N_3, \ldots, N_2$, be the minimum cost given termination is at N_2.

$G_i(x)$ is a quadratic function with coefficients, call them $p_1(i)$, $q_1(i)$, and $r_1(i)$, satisfying the usual formulas for a deterministic linear dynamics, quadratic criterion problem. $H_i(x)$ is likewise, with coefficients $p_2(i)$, $q_2(i)$, and $r_2(i)$.

For $i = 0, \ldots, N_3 - 2$,

$$F_i(x) = \min_y \left[x^2 + y^2 + F_{i+1}(x + y) \right].$$

$$F_{N_3 - 1}(x) = \min_y \left[x^2 + y^2 + v \left\{ p_1(N_3)(x + y)^2 + q_1(N_3)(x + y) + r_1(N_3) \right\} \right.$$
$$\left. + (1 - v) \left\{ p_2(N_3)(x + y)^2 + q_2(N_3)(x + y) + r_2(N_3) \right\} \right].$$

Hence $F_i(x)$ is quadratic with coefficients $u(i)$, $v(i)$, and $w(i)$ satisfying the usual formulas and

$$u(N_3) = vp_1(N_3) + (1 - v)p_2(N_3),$$
$$v(N_3) = vq_1(N_3) + (1 - v)q_2(N_3),$$
$$w(N_3) = vr_1(N_3) + (1 - v)r_2(N_3).$$

14.6. Results (14.3)–(14.5) are unaffected by the change to $\text{var}(z(i)) = \sigma^2 y^2(i)$. Likewise for (14.12)–(14.15). In (14.16), σ^2 is replaced by $\sigma^2 y^2$. Hence

$$y = - \frac{2pghx + 2ph\bar{z} + qh}{2(c + ph^2 + p\sigma^2)}.$$

Therefore

$$p(i) = a + p(i + 1)g^2 - \frac{p^2(i + 1)h^2 g^2}{c + p(i + 1)h^2 + p(i + 1)\sigma^2},$$

$$q(i) = 2p(i + 1)g\bar{z} + q(i + 1)g - \frac{p(i + 1)gh[2p(i + 1)h\bar{z} + q(i + 1)h]}{c + p(i + 1)h^2 + p(i + 1)\sigma^2},$$

$$r(i) = p(i + 1)\bar{z}^2 + q(i + 1)\bar{z} + r(i + 1) - \frac{[2p(i + 1)h\bar{z} + q(i + 1)h]^2}{4[c + p(i + 1)h^2 + p(i + 1)\sigma^2]}.$$

14.7. Let $F_i(x(i), y(i - 1)) =$ the minimum expected cost of the remaining process starting stage i in state $x(i)$ with the decision at stage $i - 1$ having been $y(i - 1)$.

$$F_i(x(i), y(i - 1)) = \min_{y(i)} \underset{z(i)}{E} \left[x^2(i) + y^2(i) + F_{i+1}(gx(i) + hy(i - 1) + z(i), y(i)) \right].$$

$$F_{N-1}(x(N - 1), y(N - 2)) = x^2(N - 1) + \underset{z(N-1)}{E} \left[(gx(N - 1) + hy(N - 2) + z(N - 1))^2 \right]$$
$$= x^2(N - 1) + g^2 x^2(N - 1) + h^2 y^2(N - 2)$$
$$+ \sigma^2 + 2ghx(N - 1)y(N - 2).$$

Assume, as an inductive hypothesis, that

$$F_{i+1}(x(i+1), y(i)) = p(i+1)x^2(i+1) + q(i+1)x(i+1)y(i) + r(i+1)y^2(i) + s(i+1).$$

$$F_i(x(i), y(i-1)) = \min_{y(i)} \, \underset{z(i)}{E} \, [x^2(i) + y^2(i) + p(i+1)(gx(i) + hy(i-1) + z(i))^2$$

$$+ q(i+1)(gx(i) + hy(i-1) + z(i))y(i) + r(i+1)y^2(i) + s(i+1)]$$

$$= \min_{y(i)} [x^2(i) + y^2(i) + p(i+1)(gx(i) + hy(i-1))^2 + p(i+1)\sigma^2$$

$$+ q(i+1)(gx(i) + hy(i-1))y(i) + r(i+1)y^2(i) + s(i+1)].$$

The minimizing y is given by

$$y(i) = -\frac{q(i+1)(gx(i) + hy(i-1))}{2(1 + r(i+1))}.$$

$$F_i(x(i), y(i-1)) = x^2(i) + \frac{q^2(i+1)(gx(i) + hy(i-1))^2}{4(1 + r(i+1))^2} + p(i+1)(gx(i) + hy(i-1))^2$$

$$+ p(i+1)\sigma^2 - \frac{q^2(i+1)(gx(i) + hy(i-1))^2}{2(1 + r(i+1))}$$

$$+ r(i+1)\frac{q^2(i+1)(gx(i) + hy(i-1))^2}{4(1 + r(i+1))^2} + s(i+1)$$

$$= x^2(i) + \frac{q^2(i+1)(gx(i) + hy(i-1))^2}{4(1 + r(i+1))}$$

$$+ p(i+1)(gx(i) + hy(i-1))^2 + p(i+1)\sigma^2 + s(i+1)$$

$$= \left(1 + \frac{q^2(i+1)g^2}{4(1 + r(i+1))} + p(i+1)g^2\right)x^2(i)$$

$$+ \left(\frac{q^2(i+1)gh}{2(1 + r(i+1))} + 2p(i+1)gh\right)x(i)y(i-1)$$

$$+ \left(\frac{q^2(i+1)h^2}{4(1 + r(i+1))} + p(i+1)h^2\right)y^2(i-1) + p(i+1)\sigma^2 + s(i+1).$$

This confirms the inductive hypothesis and yields recurrence relations for p, q, r, and s. The answer is $F_1(x(1), y(0))$.

Chapter 15.

15.1. Let B = heads, A_1 = coin 1 was flipped, A_2 = coin 2 was flipped.

$$P(A_1|B) = \frac{\frac{3}{4} \cdot \frac{1}{2}}{\frac{3}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2}} = \frac{3}{4}.$$

15.2.

$$P(A_1|B) = \frac{\frac{3}{4} \cdot \frac{1}{3}}{\frac{3}{4} \cdot \frac{1}{3} + \frac{1}{4} \cdot \frac{2}{3}} = \frac{3}{5}.$$

15.3. Let x = the event that p is between x and $x + dx$, y = the event that k flips of the coin yielded l heads.

$$f(x|y) = \frac{\binom{k}{l}x^l(1 - x)^{k-l}}{\binom{k}{l}\int_0^1 z^l(1 - z)^{k-l} \, dz} = \frac{x^l(1 - x)^{k-l}}{\frac{1}{k+1} \cdot \frac{1 \cdot 2 \cdots (k-l)}{(l+1) \cdots (k)}},$$

where the integral in the denominator is evaluated by means of $k - l$ integrations by parts.

$$\bar{p} = \int_0^1 x f(x|y)\, dx = \frac{\displaystyle\int_0^1 x^{l+1}(1-x)^{k-l}\, dx}{\displaystyle\int_0^1 z^l(1-z)^{k-l}\, dz} = \frac{\dfrac{1}{k+2} \cdot \dfrac{1 \cdot 2 \cdots (k-l)}{(l+2)\cdots(k+1)}}{\dfrac{1}{k+1} \cdot \dfrac{1 \cdot 2 \cdots (k-l)}{(l+1)\cdots(k)}} = \frac{l+1}{k+2},$$

where the integral in the numerator is evaluated by means of $k - l + 1$ integrations by parts.

15.4. For decision 1, if k_1 choices have led to l_1 upward transitions,

$$\text{Prob}\left(p_1 = \tfrac{1}{4}\,|\,k_1, l_1\right) = \frac{\left(\tfrac{1}{4}\right)^{l_1}\left(\tfrac{3}{4}\right)^{k_1-l_1}}{\left(\tfrac{1}{4}\right)^{l_1}\left(\tfrac{3}{4}\right)^{k_1-l_1} + \left(\tfrac{3}{4}\right)^{l_1}\left(\tfrac{1}{4}\right)^{k_1-l_1}},$$

$$\text{Prob}\left(p_1 = \tfrac{3}{4}\,|\,k_1, l_1\right) = \frac{\left(\tfrac{3}{4}\right)^{l_1}\left(\tfrac{1}{4}\right)^{k_1-l_1}}{\left(\tfrac{1}{4}\right)^{l_1}\left(\tfrac{3}{4}\right)^{k_1-l_1} + \left(\tfrac{3}{4}\right)^{l_1}\left(\tfrac{1}{4}\right)^{k_1-l_1}}.$$

Similarly for p_2. Let $S(k_1, l_1, k_2, l_2)$ be defined by (15.6) and i and j by (15.5).

$$S(k_1, l_1, k_2, l_2) = \min \begin{bmatrix} 1: & \text{Prob}\left(p_1 = \tfrac{1}{4}\,|\,k_1, l_1\right)\left\{\tfrac{1}{4}[a_u(i,j) + S(k_1+1, l_1+1, k_2, l_2)]\right. \\[4pt] & \left. + \tfrac{3}{4}[a_d(i,j) + S(k_1+1, l_1, k_2, l_2)]\right\} \\[4pt] & + \text{Prob}\left(p_1 = \tfrac{3}{4}\,|\,k_1, l_1\right)\left\{\tfrac{3}{4}[a_u(i,j) + S(k_1+1, l_1+1, k_2, l_2)]\right. \\[4pt] & \left. + \tfrac{1}{4}[a_d(i,j) + S(k_1+1, l_1, k_2, l_2)]\right\} \\[8pt] 2: & \text{Prob}\left(p_2 = \tfrac{1}{4}\,|\,k_2, l_2\right)\left\{\tfrac{1}{4}[a_u(i,j) + S(k_1, l_1, k_2+1, l_2+1)]\right. \\[4pt] & \left. + \tfrac{3}{4}[a_d(i,j) + S(k_1, l_1, k_2+1, l_2)]\right\} \\[4pt] & + \text{Prob}\left(p_2 = \tfrac{3}{4}\,|\,k_2, l_2\right)\left\{\tfrac{3}{4}[a_u(i,j) + S(k_1, l_1, k_2+1, l_2+1)]\right. \\[4pt] & \left. + \tfrac{1}{4}[a_d(i,j) + S(k_1, l_1, k_2+1, l_2)]\right\} \end{bmatrix}.$$

The boundary condition is (15.8). The answer is $S(0, 0, 0, 0)$.

15.5. Let $F_1(i, j) = $ the expected cost of the process starting from (i, j) with $p_1 = \tfrac{1}{4}$, $p_2 = \tfrac{1}{4}$. Let $F_2(i, j) = $ the expected cost of the process starting from (i, j) with $p_1 = \tfrac{3}{4}$, $p_2 = \tfrac{3}{4}$. Let $F_3(i, j) = $ the minimum expected cost of the process starting from (i, j) with one p equal to $\tfrac{3}{4}$ and the other equal to $\tfrac{1}{4}$.

$$F_1(i, j) = \tfrac{1}{4}[a_u(i,j) + F_1(i+1, j+1)] + \tfrac{3}{4}[a_d(i,j) + F_1(i+1, j-1)].$$

$$F_2(i, j) = \tfrac{3}{4}[a_u(i,j) + F_2(i+1, j+1)] + \tfrac{1}{4}[a_d(i,j) + F_2(i+1, j-1)].$$

$$F_3(i, j) = \min \begin{bmatrix} \tfrac{3}{4}[a_u(i,j) + F_3(i+1, j+1)] + \tfrac{1}{4}[a_d(i,j) + F_3(i+1, j-1)] \\[4pt] \tfrac{1}{4}[a_u(i,j) + F_3(i+1, j+1)] + \tfrac{3}{4}[a_d(i,j) + F_3(i+1, j-1)] \end{bmatrix}.$$

$$F_1(5, j) = F_2(5, j) = F_3(5, j) = 0 \qquad \text{for all } j.$$

The fair price is $\left\{\tfrac{1}{4}F_1(0, 0) + \tfrac{1}{4}F_2(0, 0) + \tfrac{1}{2}F_3(0, 0)\right\} - S(0, 0, 0, 0)$.

15.6. The accept assertion, for $k' = k$ and $l' > l$, says that one should accept if it is optimal to accept when fewer of the items tested are good. For $k' < k$ and $l' = l$, it asserts that one should accept if it is optimal to accept when all additional items tested are defective. For $k' < k$ and $l' > l$, the assertion says that one should accept provided it is optimal to accept if all additional items tested are defective and if, further, some items already tested as good had been defective.

   Similar reasoning justifies the reject assertion.

15.7. Compute S for $k = n - 1$ and $l = 0, 1, \ldots, n - 1$. Then compute S for $k = n - 2$ and $l = 0, 1, \ldots, n - 2$; then $k = n - 3$, etc. Furthermore, if the optimal decision in state (k', l') is accept, one need not compute S for $l > l'$. If the optimal decision at (k', l') is accept, when computing $S(k' - 1, l')$ (if it is necessary) only the accept decision need be evaluated. If the optimal decision at (k', l') is to examine another item, for $l > l'$ the reject decision need not be evaluated.

15.9. Let $N(\mu, \sigma^2)$ denote a normally distributed random variable with mean μ and variance σ^2. If w is $N(\mu, \sigma^2)$, $aw + b$ is $N(a\mu + b, a^2\sigma^2)$. Hence $gw + hy$ is $N(g\mu + hy, g^2\sigma^2)$. If x is $N(\mu_1, \sigma_1^2)$ and y is $N(\mu_2, \sigma_2^2)$, and x and y are uncorrelated, then $x + y$ is $N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$. Hence $gw + hy + r$ is $N(g\mu + hy + \bar{r}, g^2\sigma^2 + \sigma_r^2)$. Hence

$$p(w(i + 1)|y) = \frac{1}{\sqrt{2\pi(g^2\sigma^2 + \sigma_r^2)}} \exp\left(-\frac{[w(i + 1) - (g\mu + hy + \bar{r})]^2}{2(g^2\sigma^2 + \sigma_r^2)}\right).$$

15.10. Bayes' law gives

$$p(w(i + 1)|y, z) = \frac{p(z|y, w(i + 1))p(w(i + 1)|y)}{p(z|y)}. \tag{S15.1}$$

$p(z|y, w(i + 1))$ is $N(w(i + 1), \sigma_u^2)$ and does not depend on y.
$p(w(i + 1)|y)$ is $N(g\mu + hy + \bar{r}, g^2\sigma^2 + \sigma_r^2)$.
$p(z|y)$ is $N(g\mu + hy + \bar{r}, g^2\sigma^2 + \sigma_r^2 + \sigma_u^2)$.
Hence, writing out the densities in (S15.1),

$$p(w(i + 1)|y, z) = K \exp\left(-\frac{1}{2}\left[\frac{(z - w(i + 1))^2}{\sigma_u^2} + \frac{(w(i + 1) - (g\mu + hy + \bar{r}))^2}{g^2\sigma^2 + \sigma_r^2}\right.\right.$$
$$\left.\left. - \frac{(z - (g\mu + hy + \bar{r}))^2}{g^2\sigma^2 + \sigma_r^2 + \sigma_u^2}\right]\right),$$

where

$$K = \left(\frac{2\pi\sigma_u^2(g^2\sigma^2 + \sigma_r^2)}{g^2\sigma^2 + \sigma_r^2 + \sigma_u^2}\right)^{-1/2}.$$

After squaring the exponent terms and completing the square to put the numerator of the exponent in the form $(w(i + 1) - k)^2$, we conclude that $w(i + 1)$, given y and z, is normally distributed with mean $\bar{w}(i + 1)$ given by

$$\bar{w}(i + 1) = \frac{(g\mu + hy + \bar{r})\sigma_u^2 + z(g^2\sigma^2 + \sigma_r^2)}{g^2\sigma^2 + \sigma_r^2 + \sigma_u^2}$$

and variance $\sigma_{w(i+1)}^2$ given by

$$\sigma_{w(i+1)}^2 = \frac{\sigma_u^2(g^2\sigma^2 + \sigma_r^2)}{g^2\sigma^2 + \sigma_r^2 + \sigma_u^2}.$$

Since the variance is unaffected by the decision y, this is an example of passive learning. We learn, but do not choose our decision in order to learn.

INDEX