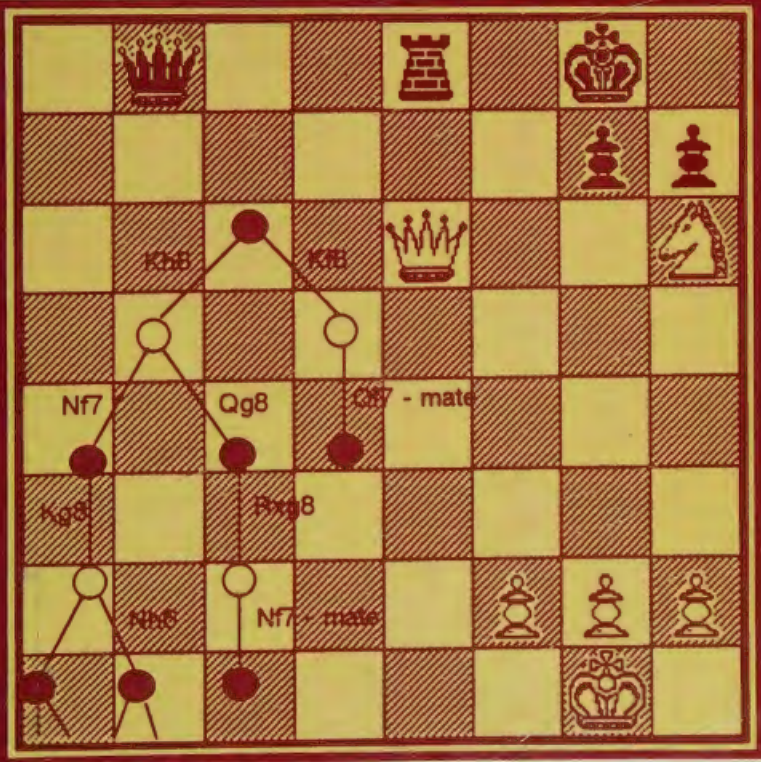
DIMITRI P. BERTSEKAS

DYNAMIC PROGRAMMING



DETERMINISTIC AND STOCHASTIC MODELS

Dynamic Programming:

Deterministic and Stochastic Models

DIMITRI P. BERTSEKAS
Department of Electrical Engineering
and Computer Science
Massachusetts Institute of Technology

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

Library of Congress Cataloging-in-Publication Data

Bertsekas, Dimitri P.
Dynamic programming.

Bibliography: p.
Includes index.
1. Dynamic programming. I. Title.
T57.83.B484 1987 519.7'03 86-30285
ISBN 0-13-221581-0

Editorial/production supervision: *Raeia Maes*
Cover design: *Ben Santora*
Manufacturing buyer: *Rhett Conklin*

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-221581-0 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

Preface

This book evolved from teaching a course on Dynamic Programming and Stochastic Control over a fourteen-year period at Stanford University, the University of Illinois, and the Massachusetts Institute of Technology. The purpose of the book is to provide a unified treatment of the subject suitable for a broad audience from engineering, operations research, and, to some extent, economics and applied mathematics. Thus, for example, we treat simultaneously stochastic control problems popular in modern control theory, Markovian decision problems popular in operations research, and a number of combinatorial problems usually addressed in computer science courses. The theory is illustrated through a large variety of examples, many of them involving applications that are important in their own right. These examples can be covered in class independently of one another, so an instructor can tailor a course to his/her audience by emphasizing the appropriate set of applications.

The mathematical prerequisite for the text is a good knowledge of introductory probability and undergraduate mathematics. This includes the equivalent of a one-semester first course in probability theory together with the usual calculus, real analysis, vector-matrix algebra, and elementary optimization theory almost all undergraduates are exposed to by their fourth year of studies. A summary of this material is provided in the appendixes. While prior courses or background on dynamic system theory, optimization, or control will undoubtedly be helpful to the reader, it is felt that the material in the text is reasonably self-contained.

Dynamic programming is a conceptually simple technique that can be

adequately explained using elementary analysis. Yet a mathematically rigorous treatment of general, stochastic dynamic programming requires the complicated machinery of measure-theoretic probability. My choice has been to bypass the complicated mathematics by carrying out the analysis in a general setting while claiming rigor only when the underlying probability spaces are countable. A mathematically rigorous treatment of the subject is carried out in my monograph "Stochastic Optimal Control: The Discrete Time Case," Academic Press, 1978, coauthored with Steven Shreve. This monograph complements the present text and provides a solid foundation for the subjects developed somewhat informally here.

I am thankful to a number of individuals and institutions for their contributions to the book. My understanding of the subject was sharpened while I worked with Steven Shreve on our 1978 monograph. Several proofs and results dealing with infinite horizon problems were improved during that time, and they are now part of the present text. Michael Caramanis, Lennart Ljung, and John Tsitsiklis taught from versions of the book and contributed several substantive comments and homework problems. I had the benefit of interaction with several able teaching assistants over the years and in this connection I would like to mention Paris Canellakis, Panos Constantopoulos, and John Tsitsiklis. A number of colleagues contributed valuable insights and information, particularly David Castanon and Krishna Pattipati. NSF supported the research on infinite horizon problems reported in Chapter 5. MIT, with its stimulating teaching and research environment, was an ideal setting for carrying out this project.

Dimitri P. Bertsekas

Life can only be understood going backwards, but it must be lived going forwards.

Kierkegaard

CHAPTER ONE

The Dynamic Programming Algorithm

1.1 THE BASIC PROBLEM

This text looks at situations where decisions are made in stages. The outcome of each decision is not fully predictable but can be observed before the next decision is made. The objective is to minimize a certain cost—a mathematical expression of what is considered desirable outcome.

A key aspect of such problems is that decisions cannot be viewed in isolation since one must balance the desire for low present cost with the possibility of high future costs being inevitable. This idea is captured in the dynamic programming technique whereby at each stage one selects a decision that minimizes the sum of the current stage cost, and the best cost that can be expected from future stages.

A very wide class of problems can be treated in this way and in this text we make an effort to keep the main ideas uncluttered by irrelevant assumptions on problem structure. To this end we formulate in this section a broadly applicable model of optimal control of a dynamic system over a finite number of stages (a finite horizon). This model will occupy us for the first four chapters; its infinite horizon version will be the subject of the last three chapters.

Two main features of the basic problem determine its structure: (1) an underlying *discrete-time dynamic system*, and (2) a *cost functional that is additive over time*. The dynamic system is of the form

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1,$$

where

k indexes discrete time,

x_k is the state of the system and summarizes past information that is relevant for future optimization,

u_k is the control or decision variable to be selected at time k with knowledge of the state x_k,

w_k is a random parameter (also called disturbance or noise),

N is the horizon or number of times control is applied.

The cost functional is additive in the sense that a cost $g_k(x_k, u_k, w_k)$ is incurred at each time k, and the total cost along any system sample trajectory is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k),$$

where $g_N(x_N)$ is a terminal cost incurred at the end of the process. However, because of the presence of w_k, cost is generally a random variable and cannot be meaningfully optimized. We therefore formulate the problem as one whereby we wish to select controls $u_0, u_1, \ldots, u_{N-1}$ so as to minimize the *expected* cost

$$E\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)\},$$

where the expectation is taken with respect to the joint distribution of the random variables involved.

A more precise definition of the terminology just used will be given shortly. We first provide some orientation by means of examples.

Inventory Control Example

Consider a problem of ordering a quantity of a certain item at the beginning of each of N time periods so as to meet a stochastic demand. Let us denote

x_k stock available at the beginning of the kth period,

u_k stock ordered (and immediately delivered) at the beginning of the kth period,

w_k demand during the kth period with given probability distribution.

We assume that $w_0, \ldots, w_{N-1}$ are independent random variables and that excess demand is backlogged and filled as soon as additional inventory becomes available. Thus stock evolves according to the discrete-time (or difference) equation

$$x_{k+1} = x_k + u_k - w_k,$$

where negative stock corresponds to backlogged demand (see Figure 1.1).

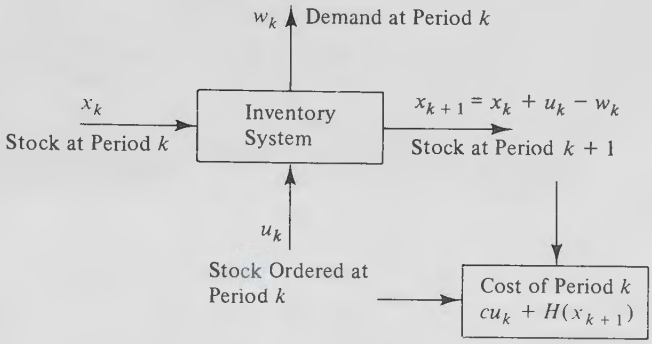


Figure 1.1 Inventory control example. The stock (state) x_k at period k, the stock ordered (control) u_k at period k, and the demand (random disturbance) w_k at period k determine the stock at the next period $k + 1$ and the cost of the kth period using the difference equation $x_{k+1} = x_k + u_k - w_k$.

The cost incurred at each period k consists of two components: (1) the purchasing cost cu_k, where c is cost per unit ordered, and (2) a cost $H(x_{k+1})$ representing a penalty for either positive stock $x_{k+1} > 0$ at the end of the period (holding cost for excess inventory) or negative stock $x_{k+1} < 0$ (shortage cost for unfilled demand). Using the equation $x_{k+1} = x_k + u_k - w_k$, we can write the cost for period k as

$$cu_k + H(x_k + u_k - w_k)$$

and the total expected cost over N periods as

$$E\left\{\sum_{k=0}^{N-1} cu_k + H(x_k + u_k - w_k)\right\}.$$

Our objective is to minimize this cost by proper choice of the orders $u_0, \ldots, u_{N-1}$ subject to the natural constraint $u_k \geq 0$, $k = 0, \ldots, N - 1$. One possibility would be to choose at time 0 all the orders $u_0, \ldots, u_{N-1}$ without waiting to see subsequent levels of demand. However, a clearly better choice would be to postpone ordering of u_k until time k when the current stock level x_k will be known. This mode of operation involves information gathering and sequential decision making based on information as it becomes available and is of central importance in dynamic programming. It implies that we are not really interested in selecting optimal numerical values for inventory orders, but rather we are interested in finding *an optimal rule for choosing at each period k an order u_k for each possible value of stock x_k that can occur*. This is an "action versus strategy" distinction. Mathematically, the problem is one of finding a sequence of *functions* μ_k, $k = 0, \ldots, N - 1$, mapping stock x_k into order u_k so as to minimize the total expected cost. The meaning of μ_k is that, for each k and

each possible value of x_k,

$$\mu_k(x_k) = \text{amount that should be ordered at time } k \text{ if stock is } x_k.$$

The sequence $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ will also be referred to as a *control law* or a *policy*. For each such π, the corresponding cost for a fixed initial stock x_0 is

$$J_\pi(x_0) = E\left\{\sum_{k=0}^{N-1} c\mu_k(x_k) + H[x_k + \mu_k(x_k) - w_k]\right\},$$

and our objective will be to minimize $J_\pi(x_0)$ for fixed x_0 over all admissible π. This is a typical dynamic programming problem. We will show in Section 2.2 that, for a reasonable choice of the cost function H, the optimal ordering rule is of the form

$$\mu_k(x_k) = \begin{cases} S_k - x_k, & \text{if } x_k < S_k, \\ 0, & \text{if } x_k \geq S_k, \end{cases}$$

where S_k is a suitable threshold level determined by the data of the problem. In other words, when stock falls below the threshold S_k, order just enough to bring stock up to S_k.

The preceding example illustrates the main ingredients of the basic problem formulation:

1. *A discrete-time system* of the form
$$x_{k+1} = f_k(x_k, u_k, w_k),$$
where f_k is some function; in this example $f_k(x_k, u_k, w_k) = x_k + u_k - w_k$.
2. *Independent random parameters* w_k. This will be generalized by allowing the probability distribution of w_k to depend on x_k and u_k; in the context of the example we can think of a situation where the level of demand w_k is influenced by the current stock level.
3. *A control constraint;* in the example $u_k \geq 0$. In general, the constraint set will depend on x_k and the time index k, that is, $u_k \in U_k(x_k)$. To see how constraints dependent on x_k can arise in the inventory context, think of a situation where there is an upper bound B on the level of stock that can be accommodated, so $u_k \leq B - x_k$.
4. *An additive cost* of the form
$$E\left\{g_N(x_N) + \sum_{k\ 0}^{N-1} g_k(x_k, u_k, w_k)\right\},$$
where g_k, $k = 0, \ldots, N$, are some functions; in the preceding example $g_N(x_N) = 0$, and $g_k(x_k, u_k, w_k) = cu_k + H(x_k + u_k - w_k)$.
5. *Optimization over control laws,* that is, rules for choosing u_k for each k and possible value of x_k.

In the preceding example, the state x_k was a real number. In other cases the state is an n-dimensional vector. It is also possible, however,

that the state takes values from a discrete set, such as the integers, or even a finite set.

A version of the inventory problem where a discrete viewpoint is more natural arises when stock is measured in whole units (such as cars), each of which is a significant fraction of x_k, u_k, or w_k. It is more appropriate then to take as state space the set of all integers, rather than the set of real numbers. The form of the system equation and the cost per period will, of course, stay the same.

In other systems the state is naturally discrete and there is no continuous counterpart of the problem. Such systems are often conveniently specified in terms of the probabilities of transition between the states. What we need to know is $p_{ij}(u, k)$ defined as the probability at time k that the next state x_{k+1} will be j, given that the current state x_k is i, and the control u_k selected is u; that is,

$$p_{ij}(u, k) = P\{x_{k+1} = j \mid x_k = i, u_k = u\}.$$

[If the system is stationary, i.e. the previous probabilities do not depend on k, we will suppress the argument k and write $p_{ij}(u)$ in place of $p_{ij}(u, k)$.] Such a system can be described alternatively in terms of a discrete-time system equation of the form

$$x_{k+1} = w_k,$$

where the probability distribution of the random parameter w_k is

$$P\{w_k = j \mid x_k = i, u_k = u\} = p_{ij}(u, k).$$

Depending on the situation at hand, it may be preferable to use a system description in terms of a difference equation or in terms of transition probabilities. We illustrate these ideas with an example.

Queueing Example

Consider a queueing system with room for n customers operating over N time periods (see Figure 1.2). We assume that service of a customer can start (end) only at the beginning (end) of a period. The probability p_m of m customers arriving during a time period is given, and the numbers of arrivals in two different periods are independent. Customers finding the system full depart without attempting to enter later. The system offers two kinds of service, *fast* and *slow*, with cost per period c_f and c_s, respectively. Service can be switched between fast and slow at the beginning of each period. If fast (slow) service is provided during a certain period, a customer in service at the beginning of the period will terminate service at the end of the period with probability q_f (respectively, q_s) independently of the number of periods the customer has been in service and the number of customers in the system ($q_f > q_s$). There is a cost $c(i)$ for each period for which there are i customers in the system. There is also a terminal cost $C(i)$ for i customers left in the system at the end of the last period. The

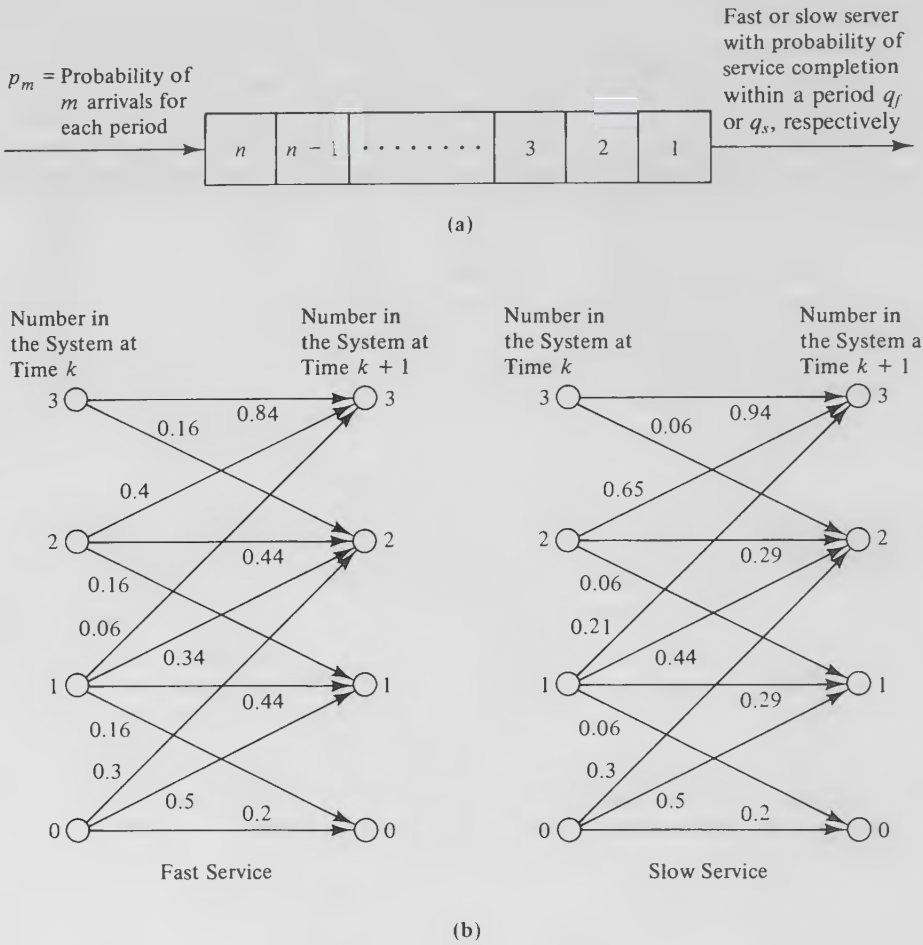


Figure 1.2 Queueing system with room for n customers. The service can be switched between fast and slow at any time period so as to minimize the sum of customer waiting and service costs: (a) Queueing system with room for n customers and two kinds of service. (b) Transition probability graphs for fast and slow service. The data assumed are $n = 3$, $p_0 = 0.2$, $p_1 = 0.5$, $p_2 = 0.3$, $p_m = 0$ for $m > 2$, and $q_f = 0.8$, $q_s = 0.3$.

problem is to choose the kind of service provided at each time period as a function of the number of customers in the system at the start of the period so as to minimize the expected total cost over N periods.

It is appropriate to take as state here the number i of customers in the system at the start of a period and as decision variable (control) the kind of service provided. The cost per period then is $c(i)$ plus c_f or c_s depending on whether fast or slow service is provided. We derive the

transition probabilities of the system. When the system is empty at the start of a period, the probability that the next state is j is independent of the kind of service provided. It equals the given probability of j customer arrivals when $j < n$

$$p_{0j}(u_f) = p_{0j}(u_s) = p_j, \qquad j = 0, 1, \ldots, n-1,$$

and it equals the probability of n or more customer arrivals when $j = n$:

$$p_{0n}(u_f) = p_{0n}(u_s) = \sum_{m=n}^{\infty} p_m.$$

When there is at least one customer in the system ($i > 0$), we have

$$\begin{aligned}
p_{ij}(u_f) &= 0, \qquad \text{if } j < i-1,\\
p_{i(i-1)}(u_f) &= q_f p_0,\\
p_{ij}(u_f) &= P\{j-i+1 \text{ arrivals, service completed}\}\\
&\quad + P\{j-i \text{ arrivals, service not completed}\}\\
&= q_f p_{j-i+1} + (1-q_f)p_{j-i}, \qquad \text{if } i-1 < j < n-1,\\
p_{i(n-1)}(u_f) &= q_f \sum_{m=n-i}^{\infty} p_m + (1-q_f)p_{n-1-i},\\
p_{in}(u_f) &= (1-q_f) \sum_{m=n-i}^{\infty} p_m.
\end{aligned}$$

The transition probabilities when slow service is provided are also given by these formulas with u_f and q_f replaced by u_s and q_s, respectively.

Transition probabilities are sometimes shown on a graph whose arcs represent transitions between various states. This is known as the *transition probability graph*, or simply *transition graph*, and is illustrated in Figure 1.2 for the special case where $n = 3$, $p_0 = 0.2$, $p_1 = 0.5$, $p_2 = 0.3$, $p_m = 0$ for $m > 2$, and $q_f = 0.8$, $q_s = 0.3$.

In our subsequent formulation we will assume that the state x_k takes values from some set S_k called the *state space*. We will not require that S_k be a finite set or a space of n-dimensional vectors. A surprising aspect of dynamic programming is that its applicability depends very little on the nature of the state space S_k (although its effectiveness certainly does depend on S_k). For this reason we find it convenient to proceed without imposing any assumptions on S_k; indeed, such assumptions would become a serious impediment later. We similarly allow u_k and w_k to take values from some unspecified spaces C_k and D_k, respectively.

Basic Problem

We are given the discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1, \tag{1.1}$$

where the state x_k is an element of a space S_k, the control u_k is an element of a space C_k, and the random "disturbance" w_k is an element of a space D_k. The control u_k is constrained to take values from a given nonempty subset $U_k(x_k)$ of C_k, which depends on the current state x_k [$u_k \in U_k(x_k)$ for all $x_k \in S_k$ and k]. The random disturbance w_k is characterized by a probability measure $P_k(\cdot|x_k, u_k)$ that may depend explicitly on x_k and u_k but not on values of prior disturbances $w_{k-1}, \ldots, w_0$. We consider the class of control laws (also called policies) that consist of a sequence of functions $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$, where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$, and is such that $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Such control laws will be termed *admissible*.

Given an initial state x_0, the problem is to find an admissible control law $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ that minimizes the cost functional

$$J_\pi(x_0) = \underset{\substack{w_k \\ k=0,\ldots,N-1}}{E} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(x_k), w_k] \right\} \tag{1.2}$$

subject to the system equation constraint

$$x_{k+1} = f_k[x_k, \mu_k(x_k), w_k], \qquad k = 0, 1, \ldots, N-1. \tag{1.3}$$

The cost functions g_k, $k = 0, 1, \ldots, N$, are given.

For a given initial state x_0, an optimal control law π^* is one that minimizes the corresponding cost

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0),$$

where Π is the set of all admissible control laws. The optimal cost corresponding to x_0 will be denoted $J^*(x_0)$; that is,

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0).$$

We view J^* as a function that assigns to each initial state x_0 the optimal cost $J^*(x_0)$ and call it the *optimal cost function* or *optimal value function*.

[For the benefit of the mathematically oriented reader we note that in the preceding equation min denotes the greatest lower bound (or infimum) of the set of numbers $\{J_\pi(x_0) \mid \pi \in \Pi\}$. A notation more in line with normal mathematical usage would be to write $J^*(x_0) = \inf_{\pi \in \Pi} J_\pi(x_0)$. However (as discussed in Appendix B), we find it convenient to use *min* in place of *inf* even when the infimum is not attained. It is less distracting and will not lead to any confusion.]

Role of Information in the Basic Problem

We mentioned earlier that a policy $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ may be viewed as a plan that specifies the control to be applied at each time for every state that may occur at that time. It is important to realize that this mode of operation implies *information gathering*. The information received by

the controller is the value of the current state at each time and is utilized directly during the control process, since the control at time k depends on the current state x_k via the function μ_k (cf. Figure 1.3). The effects of the availability of this information may be significant indeed. If this information is not available, the controller cannot adapt appropriately to unexpected values of the state, and as a result the cost can be adversely affected. For example, in the inventory control problem considered earlier, the information that becomes available at the beginning of each period k is the inventory stock x_k. Clearly, this information is very important to the inventory manager, who will want to adjust the amount u_k to be purchased depending on whether the current stock x_k is running high or low.

Note, however, that whereas availability of the state information cannot hurt, it may not result in an advantage either. For instance, in deterministic control problems, where no random disturbances are present, one can predict the future states given the initial state and the sequence of controls. Therefore, optimization over all sequences $\{u_0, u_1, \ldots, u_{N-1}\}$ of controls leads to the same optimal cost as optimization over all admissible policies. The same fact may be true even in some stochastic control problems (see Problem 13). This brings up a related issue. Assuming no information is forgotten, the controller actually knows the prior states and controls x_0, $u_0, \ldots, x_{k-1}, u_{k-1}$, as well as the current state x_k. Therefore, the question arises whether policies that use the entire system history can be superior to policies that use just the current state. The answer turns out to be negative (see [B23]). The intuitive reason is that, for a given problem, time k and state x_k, all future expected costs depend explicitly just on x_k and not on prior history.

Theoretical Limitations of the Formulation of the Basic Problem

Before proceeding with the development of the dynamic programming algorithm, we try to clarify certain aspects of our problem that do not lie on firm mathematical ground. The issue here is one of mathematical rigor and is highly technical in nature. The reader who is not mathematically

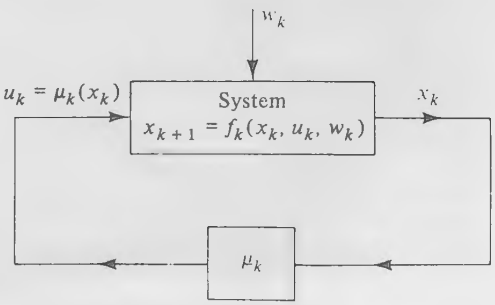


Figure 1.3 Information gathering in the basic problem. At each time k the controller observes the current state x_k and applies control $u_k = \mu_k(x_k)$ that depends on that state.

inclined need not be concerned about it and can skip the rest of this section without loss of continuity.

First, once an admissible control law $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ is adopted, the following sequence of events is envisioned for each stage $k = 0, 1, \ldots, N-1$:

1. The controller observes x_k and applies $u_k = \mu_k(x_k)$.
2. The disturbance w_k is generated according to the given probability measure $P_k(\cdot|x_k, \mu_k(x_k))$.
3. The cost $g_k[x_k, \mu_k(x_k), w_k]$ is incurred and added to previous costs.
4. The next state x_{k+1} is generated according to the system equation

$$x_{k+1} = f_k[x_k, \mu_k(x_k), w_k].$$

If this is the last stage ($k = N-1$), the terminal cost $g_N(x_N)$ is added to previous costs and the process terminates. Otherwise, k is incremented, and the same sequence of events is repeated for the next stage.

This process is well defined and couched in precise probabilistic terms. Things are complicated, however, by the need to view the cost

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(x_k), w_k]$$

as a *well-defined random variable* with well-defined expected value. The framework of probability theory requires that for each $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ we define an underlying probability space, that is, a set Ω, a collection of events in Ω, and a probability measure on these events. Furthermore, the cost must be a well-defined random variable on this space in the sense of Appendix C (a measurable function from the probability space into the real line in the terminology of measure-theoretic probability theory). For this to be true, additional (measurability) assumptions on the functions f_k, g_k, and μ_k may be required, and it may be necessary to introduce additional structure on the spaces S_k, C_k, and D_k. Furthermore, these assumptions may restrict the class of admissible control laws since the functions μ_k may be constrained to satisfy additional (measurability) requirements.

Thus, unless these additional assumptions and structure are specified, the problem is formulated inadequately. On the other hand, a rigorous formulation of the basic problem for general state, control, and disturbance spaces is well beyond the mathematical framework of this introductory text and will not be undertaken here (see [B23]). Nonetheless, these difficulties are mainly technical and do not substantially affect the basic results to be obtained. For this reason we find it convenient to proceed with informal derivations and arguments in much the same way as in all introductory texts and most journal literature on the subject.

We would like to stress, however, that under the assumption that *the*

disturbance spaces D_k, $k = 0, 1, \ldots, N-1$, *are countable sets* all the mathematical difficulties mentioned disappear since, for this case, *with the only additional assumption that the expected values of all terms in the cost* (1.2) *exist and are finite for every admissible policy* π, one can provide a sound framework for the problem.

One easy way to do this when D_k are countable is to rewrite all expected values in the cost as infinite sums in terms of the probabilities of the elements of D_k. Another way is to write the cost $J_\pi(x_0)$ as

$$J_\pi(x_0) = \underset{x_1,\ldots,x_N}{E}\left\{g_N(x_N) + \sum_{k=0}^{N-1} \bar{g}_k[x_k, \mu_k(x_k)]\right\}, \tag{1.4}$$

where

$$\bar{g}_k[x_k, \mu_k(x_k)] = \underset{w_k}{E}\{g_k[x_k, \mu_k(x_k), w_k] \mid x_k, \mu_k(x_k)\},$$

with the preceding expectation taken with respect to the probability distribution $P_k(\cdot|x_k, \mu_k(x_k))$ defined on the countable set D_k. Then one may take as the basic probability space the Cartesian product of $\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_N$, where

$$\begin{aligned}
\bar{S}_1 &= \{x_1 \in S_1 \mid x_1 = f_0[x_0, \mu_0(x_0), w_0], w_0 \in D_0\},\\
\bar{S}_{k+1} &= \{x_{k+1} \in S_{k+1} \mid x_{k+1} = f_k[x_k, \mu_k(x_k), w_k],\\
&\qquad x_k \in \bar{S}_k, w_k \in D_k\}, \qquad k = 1, 2, \ldots, N-1.
\end{aligned}$$

The set $\bar{S}_k$ is the subset of S_k of all states that can be reached at time k when the control law $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ is employed. The fact that $D_0, D_1, \ldots, D_{N-1}$ are countable sets ensures that the sets $\bar{S}_1, \ldots, \bar{S}_N$ are also countable (this is true since the union of any countable collection of countable sets is a countable set). Now the system equation (1.3), the probability distributions $P_k(\cdot|x_k, \mu_k(x_k))$, the initial state x_0, and the control law $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ define a probability distribution on the countable set $\bar{S}_1 \times \bar{S}_2 \times \cdots \times \bar{S}_N$, and the expectation in (1.4) is defined with respect to this latter distribution.

In conclusion, the basic problem has been formulated rigorously only when the disturbance spaces $D_0, \ldots, D_{N-1}$ are countable sets. In the absence of countability of D_k, the reader should interpret subsequent results and conclusions as essentially correct but mathematically imprecise statements. In fact, when discussing infinite horizon problems (where the need for precision is greater), we will make the countability assumption explicit. We note, however, that the advanced reader will have little difficulty in establishing rigorously most of our subsequent results concerning specific applications in Chapters 2 and 3. This can be done as explained in the Notes to this chapter and in Problem 12.

1.2 THE DYNAMIC PROGRAMMING ALGORITHM

The dynamic programming (DP) technique rests on a very simple idea, the *principle of optimality*. The name is due to Bellman, who contributed a great deal to the popularization of DP and to its transformation into a systematic tool. Roughly, the principle of optimality states the following rather obvious fact.

Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be an optimal control law for the basic problem. Consider the subproblem whereby we are at state x_i at time i and wish to minimize the "cost-to-go" from time i to time N;

$$E\{g_N(x_N) + \sum_{k=i}^{N-1} g_k[x_k, \mu_k(x_k), w_k]\},$$

and assume that when using π^* the state x_i occurs with positive probability. Then the truncated control law $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ is optimal for this subproblem.

The intuitive justification of the principle of optimality is very simple. If the truncated control law $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ were not optimal as stated, we would be able to reduce the cost further by switching to an optimal policy for the subproblem once we reach x_i. For an auto travel analogy, suppose we have found the fastest route from Los Angeles to Boston and this route passes through Chicago. The principle of optimality translates to the obvious fact that the Chicago to Boston portion of the route is also a fastest route for a trip that starts from Chicago and ends in Boston.

It is perhaps best to introduce the DP algorithm by means of an example.

Inventory Control Example (continued)

Consider the inventory control example of the previous section and the following procedure for determining the optimal inventory ordering policy starting with the last time period and proceeding backward in time.

$N - 1$ Period Assume that at the beginning of period $N - 1$ the stock available is x_{N-1}. Clearly, no matter what happened in the past, the inventory manager should order inventory $u_{N-1}^* = \mu_{N-1}^*(x_{N-1})$, which minimizes over u_{N-1} the sum of the ordering, holding, and shortage costs for the last time period, which is equal to

$$\underset{w_{N-1}}{E}\{cu_{N-1} + H(x_{N-1} + u_{N-1} - w_{N-1})\}.$$

Let us denote the optimal cost for the last period by $J_{N-1}(x_{N-1})$:

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1} \geq 0} \underset{w_{N-1}}{E}\{cu_{N-1} + H(x_{N-1} + u_{N-1} - w_{N-1})\}.$$

Naturally, J_{N-1} is a function of the stock x_{N-1}. It is calculated for each x_{N-1} either analytically or numerically (in which case a table is used for computer storage of the function J_{N-1}). In the process of calculating J_{N-1} we obtain the optimal inventory ordering policy $\mu_{N-1}^*(x_{N-1})$ for the last period, where $\mu_{N-1}^*(x_{N-1}) \geq 0$ minimizes the right side of the preceding equation for each value of x_{N-1}.

$N - 2$ Period Assume that at the beginning of period $N - 2$ the inventory is x_{N-2}. Now it is clear that the inventory manager should order inventory $u_{N-2} = \mu_{N-2}^*(x_{N-2})$, which minimizes not just the expected cost of period $N - 2$ but rather the

(expected cost of period $N - 2$) + (expected cost of period $N - 1$, given that an optimal policy will be used at period $N - 1$).

This, however, is equal to

$$\underset{w_{N-2}}{E}\{cu_{N-2} + H(x_{N-2} + u_{N-2} - w_{N-2})\} + \underset{w_{N-2}}{E}\{J_{N-1}(x_{N-1})\}.$$

Using the system equation $x_{N-1} = x_{N-2} + u_{N-2} - w_{N-2}$, the last term is also written $E_{w_{N-2}}\{J_{N-1}(x_{N-2} + u_{N-2} - w_{N-2})\}$.

Thus the optimal cost $J_{N-2}(x_{N-2})$ for the last two periods, given that we are at state x_{N-2}, is given by

$$J_{N-2}(x_{N-2}) = \min_{u_{N-2} \geq 0} \underset{w_{N-2}}{E}\{cu_{N-2} + H(x_{N-2} + u_{N-2} - w_{N-2}) + J_{N-1}(x_{N-2} + u_{N-2} - w_{N-2})\}.$$

Again $J_{N-2}(x_{N-2})$ is calculated for every x_{N-2}. At the same time the optimal ordering policy $\mu_{N-2}^*(x_{N-2})$ is also computed.

k Period Similarly, we have that at period k and for initial inventory x_k the inventory manager should order u_k to minimize

(expected cost of period k) + (expected cost of periods $k + 1, \ldots, N - 1$, given that an optimal policy will be used for these periods).

By denoting by $J_k(x_k)$ the optimal cost, we have

$$J_k(x_k) = \min_{u_k \geq 0} \underset{w_k}{E}\{cu_k + H(x_k + u_k - w_k) + J_{k+1}(x_k + u_k - w_k)\}, \tag{1.5}$$

which is actually the dynamic programming equation for this problem.

The functions $J_k(x_k)$ denote the optimal expected cost for the remaining periods when starting at period k and with initial inventory x_k. These functions are computed recursively backward in time, starting at period $N - 1$ and ending at period 0. The value $J_0(x_0)$ is the optimal expected cost for the process when the initial inventory at time 0 is x_0. During the calculations the optimal inventory policy, $\{\mu_0^*(x_0), \mu_1^*(x_1), \ldots, \mu_{N-1}^*(x_{N-1})\}$

is simultaneously computed from minimization of the right side of (1.5) for every x_k and k.

The example illustrates the main advantage offered by DP. Our original inventory problem requires an optimization over the set of policies, that is, the set of sequences of functions of the current stock (more generally the current state). The DP algorithm of (1.5) decomposes this problem into a sequence of minimization problems that is carried out over the set of orders (more generally the space of controls). Each of these problems is far simpler than the original.

We now state the DP algorithm for the basic problem and show its optimality.

Proposition. Let $J^*(x_0)$ be the optimal cost. Then

$$J^*(x_0) = J_0(x_0),$$

where the function J_0 is given by the last step of the following algorithm, which proceeds backward in time from period $N - 1$ to period 0:

$$J_N(x_N) = g_N(x_N) \tag{1.6}$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \underset{w_k}{E} \{g_k(x_k, u_k, w_k) + J_{k+1}[f_k(x_k, u_k, w_k)]\}, \tag{1.7}\dagger$$

$$k = 0, 1, \ldots, N - 1.$$

Furthermore, if $u_k^* = \mu_k^*(x_k)$ minimizes the right side of (1.7) for each x_k and k, the control law $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is optimal.

Proof. The fact that the probability measure characterizing w_k depends only on x_k and u_k and not on prior values of disturbances $w_0, \ldots, w_{k-1}$ allows us to write $J^*(x_0)$ in the form

$$J^*(x_0) = \min_{\mu_0, \ldots, \mu_N} \left[\underset{w_0}{E} \left\{ g_0[x_0, \mu_0(x_0), w_0] + \underset{w_1}{E} \left\{ g_1[x_1, \mu_1(x_1), w_1] + \cdots \right.\right.\right.$$

$$\left.\left.\left. + \underset{w_{N-1}}{E} \{g_{N-1}[x_{N-1}, \mu_{N-1}(x_{N-1}), w_{N-1}] + g_N(x_N)\} \cdots \right\} \right\} \right],$$

where the expectation over w_k, $k = 0, 1, \ldots, N - 1$, is conditional on x_k and $\mu_k(x_k)$. This expression may also be written

† Both the DP algorithm and its proof are, of course, rigorous only if the basic problem is rigorously formulated. As explained in the previous section, this is the case when the disturbance spaces D_k, $k = 0, 1, \ldots, N - 1$, are countable sets and the expected values of all terms in the expression of the cost functional (1.2) are well defined and finite for every admissible policy π. In addition, it is assumed that the expected value in (1.7) exists and is finite for all $u_k \in U_k(x_k)$ and all $x_k \in S_k$. We further note that, although not explicitly denoted, the expectation in (1.7) is taken with respect to the probability measure characterizing w_k, which depends on both x_k and u_k.

$$J^*(x_0) = \min_{\mu_0}\left[E_{w_0}\left\{g_0[x_0, \mu_0(x_0), w_0] + \min_{\mu_1}\left[E_{w_1}\left\{g_1[x_1, \mu_1(x_1), w_1] + \cdots\right.\right.\right.$$

$$\left.\left.\left. + \min_{\mu_{N-1}}\left[E_{w_{N-1}}\{g_{N-1}[x_{N-1}, \mu_{N-1}(x_{N-1}), w_{N-1}] + g_N(x_N)\}\right]\cdots\right\}\right]\right\}\right].$$

In this equation the minimizations are over all functions μ_k such that $\mu_k(x_k) \in U_k(x_k)$ for all x_k and k. In addition, the minimization is subject to the system equation constraint

$$x_{k+1} = f_k[x_k, \mu_k(x_k), w_k].$$

Now we use the fact that for any function F of x, u, we have

$$\min_{\mu \in M} F[x, \mu(x)] = \min_{u \in U(x)} F(x, u),$$

where M is the set of all functions $\mu(x)$ such that $\mu(x) \in U(x)$ for all x.

By applying this fact in the equation for $J^*(x_0)$, using the substitution $x_{k+1} = f_k(x_k, u_k, w_k)$, and introducing the functions J_k of (1.7), we obtain the desired result:

$$J^*(x_0) = J_0(x_0).$$

It is also clear that $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is an optimal control law if $\mu_k^*(x_k)$ minimizes the right side of (1.7) for each x_k and k, since such a control law attains the optimal cost. Q.E.D.

The argument of the preceding proof can also be used to establish an interpretation of $J_k(x_k)$. It is the optimal cost for an $(N - k)$-stage problem starting at state x_k and time k and ending at time N. We consequently call $J_k(x_k)$ the *cost-to-go* at state x_k and time k, and refer to J_k as the *cost-to-go function* at time k. Ideally, we would like to use the DP algorithm to determine closed-form expressions for J_k. Otherwise, one hopes to obtain useful characterizations of J_k or μ_k^*. In many cases one has to resort to numerical solution of the DP equations. This may be quite time consuming since the minimization in (1.7) must be carried out for each value of x_k. Typically, the state space is discretized and the minimization is carried out for a finite number of states x_k. The computational requirements are proportional to the number of discretization points. Thus for complex multidimensional problems the computational burden may be prohibitive. Nonetheless, DP is the only general approach for sequential optimization under uncertainty.

We now provide examples illustrating the analytical and computational aspects of the DP algorithm.

Example 1

A certain material is passed through a sequence of two ovens (see Figure 1.4). Denote

x_0: initial temperature of the material,

x_k, $k = 1, 2$: temperature of the material at the exit of oven k,

u_{k-1}, $k = 1, 2$: prevailing temperature in oven k.

We assume a model of the form

$$x_{k+1} = (1 - a)x_k + au_k, \qquad k = 0, 1,$$

where a is some scalar from the interval (0, 1). The objective is to get the final temperature x_2 close to a given target T, while expending relatively little energy. This is expressed by a cost function of the form

$$r(x_2 - T)^2 + u_0^2 + u_1^2,$$

where $r > 0$ is a given scalar. We assume no constraints on u_k. (In reality, there are constraints, but if we can solve the unconstrained problem and verify that the solution satisfies the constraints, everything will be fine.)

We see that this is a deterministic problem that fits the basic framework. We have $N = 2$ and a terminal cost $g_2(x_2) = r(x_2 - T)^2$, so the initial condition for the DP algorithm is [cf. (1.6)]

$$J_2(x_2) = r(x_2 - T)^2.$$

For the next-to-last stage, we have [cf. (1.7)]

$$J_1(x_1) = \min_{u_1}[u_1^2 + J_2(x_2)]$$

$$= \min_{u_1}[u_1^2 + J_2[(1 - a)x_1 + au_1]].$$

Substituting the previous form of J_2, we obtain

$$J_1(x_1) = \min_{u_1}[u_1^2 + r[(1 - a)x_1 + au_1 - T]^2]. \tag{1.8}$$

This minimization will be done by setting to zero the derivative with respect to u_1. We thus have

$$0 = 2u_1 + 2ra[(1 - a)x_1 + au_1 - T],$$

and by collecting terms we obtain the optimal temperature for the last oven:

$$u_1 = \mu_1^*(x_1) = \frac{ra[T - (1 - a)x_1]}{1 + ra^2}$$

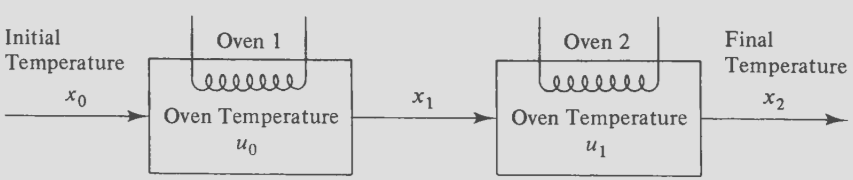


Figure 1.4 Problem of Example 1. The temperature of the material evolves according to $x_{k+1} = (1 - a)x_k + au_k$, where a is some scalar with $0 < a < 1$

Note that this is not a single control but rather a control function, a rule that tells us the optimal oven temperature u_1 for each possible state x_1.

By substituting the optimal u_1 in the expression (1.8) for J_1, we obtain

$$J_1(x_1) = \frac{r^2a^2[(1-a)x_1 - T]^2}{(1+ra^2)^2} + r\left[(1-a)x_1 + \frac{ra^2[T-(1-a)x_1]}{1+ra^2} - T\right]^2$$

$$= \frac{r^2a^2[(1-a)x_1 - T]^2}{(1+ra^2)^2} + r\left(\frac{ra^2}{1+ra^2} - 1\right)^2[(1-a)x_1 - T]^2,$$

and finally

$$J_1(x_1) = \frac{r[(1-a)x_1 - T]^2}{1+ra^2}.$$

We now go one stage back to stage 0. We have [cf. (1.7)]

$$J_0(x_0) = \min_{u_0}[u_0^2 + J_1(x_1)]$$

$$= \min_{u_0}[u_0^2 + J_1[(1-a)x_0 + au_0]],$$

and by substituting the expression already obtained for J_1, we have

$$J_0(x_0) = \min_{u_0}\left[u_0^2 + \frac{r[(1-a)^2x_0 + (1-a)au_0 - T]^2}{1+ra^2}\right].$$

We minimize with respect to u_0 by setting the corresponding derivative to zero. We obtain

$$0 = 2u_0 + \frac{2r(1-a)a[(1-a)^2x_0 + (1-a)au_0 - T]}{1+ra^2}.$$

This yields, after some calculation, the optimal temperature of the first oven:

$$u_0 = \mu_0^*(x_0) = \frac{r(1-a)a[T-(1-a)^2x_0]}{1+ra^2[1+(1-a)^2]}.$$

The optimal cost is obtained by substituting this expression in the formula for J_0. This leads to a straightforward but lengthy calculation, which in the end yields the rather simple formula

$$J_0(x_0) = \frac{r[(1-a)^2x_0 - T]^2}{1+ra^2[1+(1-a)^2]}.$$

This completes the solution of the problem.

Several noteworthy features in this example, as we will see later, admit broad generalizations. The first is the facility with which we obtained an analytical solution. A little thought while tracing the steps of the algorithm will convince the reader that what makes the easy solution possible is the quadratic nature of the cost and the linearity of the system equation. Indeed, in Section 2.1 we will see that, generally, when the system is linear and the cost is quadratic then, regardless of the number of stages N, the optimal policy admits an analytical expression.

Another noteworthy feature of this example is that the optimal policy remains unaffected when a zero-mean stochastic disturbance is added in

the system equation. To see this, assume that the material's temperature evolves according to

$$x_{k+1} = (1 - a)x_k + au_k + w_k, \qquad k = 0, 1,$$

where w_0, w_1 are independent random variables with given distribution, zero mean

$$E\{w_0\} = E\{w_1\} = 0,$$

and finite variance. Then the equation for J_1 [cf. (1.7)] becomes

$$\begin{aligned} J_1(x_1) &= \min_{u_1} E_{w_1}\{u_1^2 + r[(1 - a)x_1 + au_1 + w_1 - T]^2\} \\ &= \min_{u_1}[u_1^2 + r[(1 - a)x_1 + au_1 - T]^2 \\ &\quad + 2rE\{w_1\}[(1 - a)x_1 + au_1 - T] + rE\{w_1^2\}]. \end{aligned}$$

Therefore, using the fact that $E\{w_1\} = 0$, we obtain

$$J_1(x_1) = \min_{u_1}[u_1^2 + r[(1 - a)x_1 + au_1 - T]^2] + rE\{w_1^2\}.$$

Comparing this equation with (1.8), we see that the presence of w_1 has resulted in an additional inconsequential term, $rE\{w_1^2\}$. Therefore, the optimal policy for the last stage remains unaffected by the presence of w_1, while $J_1(x_1)$ is increased by the constant term $rE\{w_1^2\}$. It is easily seen that a similar situation also holds for the first stage. In particular, the optimal cost is given by the same expression as before except for the additional term $r(E\{w_0^2\} + E\{w_1^2\})$.

The property whereby the optimal policy is unaffected by the presence of zero-mean disturbances is a manifestation of the *certainty equivalence principle*, which holds for several types of problems involving a linear system and a quadratic cost (see Sections 2.1, 3.2, 3.3, and 6.1).

Example 2

Consider an inventory control problem similar to the one of Section 2.1 but different in that *inventory and demand are nonnegative integer variables*. Furthermore, assume that *there is an upper bound on the stock* $(x_k + u_k)$ that can be stored and also assume *that the excess demand* $(w_k - x_k - u_k)$ *is lost*. As a result, the stock equation takes the form

$$x_{k+1} = \max(0, x_k + u_k - w_k).$$

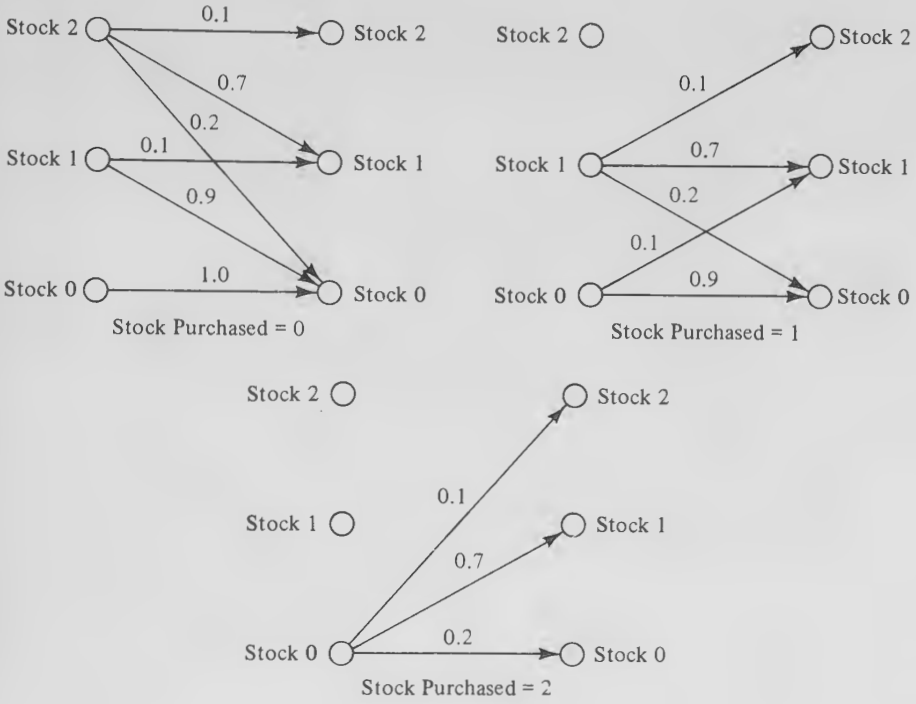
Assume that the maximum capacity $(x_k + u_k)$ for stock is 2 units, that the planning horizon N is 3 periods, and that the ordering cost c is 1 unit. The holding/shortage cost per stage is given by

$$H(x_k + u_k - w_k) = \max(0, x_k + u_k - w_k) + 3\max(0, w_k - x_k - u_k).$$

The terminal state cost is zero. The initial stock x_0 is given, and the demand w_k has the same probability distribution for all periods, given by

$$p(w_k = 0) = 0.1, \qquad p(w_k = 1) = 0.7, \qquad p(w_k = 2) = 0.2$$

The system can also be represented in terms of the probabilities of transition between the three possible states 0, 1, 2 for the different values of control (see Figure 1.5a).



(a)

	Stage 0		Stage 1		Stage 2	
Stock	Cost-to-go	Opt. Stock to Purchase	Cost-to-go	Opt. Stock to Purchase	Cost-to-go	Opt. Stock to Purchase
0	4.9	1	3.3	1	1.7	1
1	3.9	0	2.3	0	0.7	0
2	3.35	0	1.82	0	0.9	0

(b)

Figure 1.5 System and DP results for Example 2: (a) Transition probability diagrams for the different values of stock purchased (control). The numbers next to the arcs are the transition probabilities. The control $u = 1$ is not available at state 2 because of the limitation $x_k + u_k \leq 2$. Similarly, the control $u = 2$ is available only at state 0. (b) Results of the DP algorithm for Example 2.

The starting equation for the DP algorithm is

$$J_3(x_3) = 0,$$

since the terminal state cost is zero [cf. (1.6)]. The algorithm takes the form [cf. (1.7)]

$$J_k(x_k) = \min_{\substack{0 \le u_k \le 2 - x_k \\ u_k = 0,1,2}} E_{w_k} \{u_k + \max(0, x_k + u_k - w_k) + 3\max(0, w_k - x_k - u_k) + J_{k+1}[\max(0, x_k + u_k - w_k)]\}, \qquad k = 0, 1, 2,$$

where x_k, u_k, w_k can take the values 0, 1, and 2.

Stage 2 We compute $J_2(x_2)$ for each of the three possible states:

$$J_2(0) = \min_{u_2 = 0,1,2} E_{w_2} \{u_2 + \max(0, u_2 - w_2) + 3\max(0, w_2 - u_2)\}$$

$$= \min_{u_2 = 0,1,2} \{u_2 + 0.1[\max(0, u_2) + 3\max(0, -u_2)] + 0.7[\max(0, u_2 - 1) + 3\max(0, 1 - u_2)] + 0.2[\max(0, u_2 - 2) + 3\max(0, 2 - u_2)]\}.$$

We calculate the expectation of the right side for each of the three possible values of u_2:

$$u_2 = 0: \quad E\{\cdot\} = 0.7 \times 3 \times 1 + 0.2 \times 3 \times 2 = 3.3,$$

$$u_2 = 1: \quad E\{\cdot\} = 1 + 0.1 \times 1 + 0.2 \times 3 \times 1 = 1.7,$$

$$u_2 = 2: \quad E\{\cdot\} = 2 + 0.1 \times 2 + 0.7 \times 1 = 2.9.$$

Hence we have, by selecting the minimizing u_2,

$$\triangleright \qquad J_2(0) = 1.7, \qquad \mu_2^*(0) = 1 \qquad \triangleleft$$

For $x_2 = 1$, we have

$$J_2(1) = \min_{u_2 \; 0\,1} E_{w_2} \{u_2 + \max(0, 1 + u_2 - w_2) + 3\max(0, w_2 - 1 - u_2)\}$$

$$= \min_{u_2 \; 0,1} \{u_2 + 0.1[\max(0, 1 + u_2) + 3\max(0, -1 - u_2)] + 0.7[\max(0, u_2) + 3\max(0, -u_2)] + 0.2[\max(0, u_2 - 1) + 3\max(0, 1 - u_2)]\},$$

$$u_2 = 0: \quad E\{\cdot\} = 0.1 \times 1 + 0.2 \times 3 \times 1 \quad 0.7,$$

$$u_2 = 1. \quad E\{\cdot\} - 1 + 0.1 \times 2 + 0.7 \times 1 \quad 1.9.$$

Hence

$$\triangleright \qquad J_2(1) = 0.7 \qquad \mu_2^*(1) = 0 \qquad \triangleleft$$

For $x_2 = 2$, the only admissible control is $u_2 = 0$, so we have

$$J_2(2) - E_{w_2} \{\max(0, 2 - w_2) + 3\max(0, w_2 - 2)\}$$

$$= 0.1 \times 2 + 0.7 \times 1 = 0.9,$$

$$\triangleright \qquad J_2(2) - 0.9, \qquad \mu_2^*(2) = 0 \qquad \triangleleft$$

Stage 1 Again we compute $J_1(x_1)$ for each of the three possible states $x_2 = 0, 1, 2$ using the values $J_2(0)$, $J_2(1)$, $J_2(2)$ obtained in the previous stage:

$$J_1(0) = \min_{u_1=0,1,2} E_{w_1}\{u_1 + \max(0, u_1 - w_1) + 3\max(0, w_1 - u_1) + J_2[\max(0, u_1 - w_1)]\},$$

$$u_1 = 0: \quad E\{\cdot\} = 0.1 \times J_2(0) + 0.7[3 \times 1 + J_2(0)] + 0.2[3 \times 2 + J_2(0)] = 5.0,$$

$$u_1 = 1: \quad E\{\cdot\} = 1 + 0.1[1 + J_2(1)] + 0.7 \times J_2(0) + 0.2[3 \times 1 + J_2(0)] = 3.3,$$

$$u_1 = 2: \quad E\{\cdot\} = 2 + 0.1[2 + J_2(2)] + 0.7[1 + J_2(1)] + 0.2 \times J_2(0) = 3.82,$$

▷ $J_1(0) = 3.3, \qquad \mu_1^*(0) = 1,$ ◁

$$J_1(1) = \min_{u_1=0,1} E_{w_1}\{u_1 + \max(0, 1 + u_1 - w_1) + 3\max(0, w_1 - 1 - u_1) + J_2[\max(0, 1 + u_1 - w_1)]\}$$

$$u_1 = 0: \quad E\{\cdot\} = 0.1[1 + J_2(1)] + 0.7 \times J_2(0) + 0.2[3 \times 1 + J_2(0)] = 2.3,$$

$$u_1 = 1: \quad E\{\cdot\} = 1 + 0.1[2 + J_2(2)] + 0.7[1 + J_2(1)] + 0.2 \times J_2(0) = 2.82,$$

▷ $J_1(1) = 2.3, \qquad \mu_1^*(1) = 0,$ ◁

$$J_1(2) = E\{\max(0, 2 - w_1) + 3\max(0, w_1 - 2) + J_2[\max(0, 2 - w_1)]\}$$

$$= 0.1[2 + J_2(2)] + 0.7[1 + J_2(1)] + 0.2 \times J_2(0) = 1.82,$$

▷ $J_1(2) = 1.82, \qquad \mu_1^*(2) = 0.$ ◁

Stage 0 Here we need only compute $J_0(0)$ since the initial state is known to be zero. We have

$$J_0(0) = \min_{u_0=0,1,2} E_{w_0}\{u_0 + \max(0, u_0 - w_0) + 3\max(0, w_0 - u_0) + J_1[\max(0, u_0 - w_0)]\},$$

$$u_0 = 0: \quad E\{\cdot\} = 0.1 \times J_1(0) + 0.7[3 \times 1 + J_1(0)] + 0.2[3 \times 2 + J_1(0)] = 6.6,$$

$$u_0 = 1: \quad E\{\cdot\} = 1 + 0.1[1 + J_1(1)] + 0.7 \times J_1(0) + 0.2[3 \times 1 + J_1(0)] = 4.9,$$

$$u_0 = 2: \quad E\{\cdot\} = 2 + 0.1[2 + J_1(2)] + 0.7[1 + J_1(1)] + 0.2 \times J_1(0) = 5.352,$$

▷ $J_0(0) = 4.9, \qquad \mu_0^*(0) = 1.$ ◁

If the initial state were not known a priori, we would have to compute in a similar

manner $J_0(1)$ and $J_0(2)$ as well as the minimizing u_0. These calculations yield

$$\triangleright \qquad J_0(1) = 3.9, \qquad \mu_0^*(1) = 0, \qquad \triangleleft$$

$$\triangleright \qquad J_0(2) = 3.352 \qquad \mu_0^*(2) = 0 \qquad \triangleleft$$

Thus the optimal ordering policy for each period is to order one unit if the current stock is zero, and order nothing otherwise. The results of the DP algorithm are given in tabular form in Figure 1.5b.

Example 3

Finite State Systems. We mentioned earlier (cf. the queueing example in the previous section) that systems with a finite number of states can be represented either by a discrete-time system equation or in terms of the probabilities of transition between the states (cf. Figures 1.2 and 1.5). Let us work out the corresponding DP algorithm. We will assume for the sake of the following discussion that the problem is stationary (i.e. the transition probabilities, the cost per stage, and the control constraint set do not change from one stage to the next). Then, if

$$p_{ij}(u) = P\{x_{k+1} = j \mid x_k = i, u_k = u\}$$

are the transition probabilities, we can alternatively represent the system by the system equation (cf. the discussion of the previous section)

$$x_{k+1} = w_k,$$

where the probability distribution of the disturbance w_k is

$$P\{w_k = j \mid x_k = i, u_k = u\} = p_{ij}(u)$$

Using this system equation and denoting by $g(i, u)$ the expected cost per stage at state i when control u is applied, the DP algorithm can be rewritten as

$$J_k(i) = \min_{u \in U(i)} [g(i, u) + E\{J_{k+1}(w_k)\}]$$

or equivalently (in view of the distribution of w_k given previously)

$$J_k(i) = \min_{u \in U(i)} \Big[g(i, u) + \sum_j p_{ij}(u) J_{k+1}(j)\Big] \qquad k = 0, 1, \ldots, N-1$$

As an illustration, in the queueing problem of the previous section this algorithm takes the form

$$J_N(i) = C(i), \qquad i = 0, 1, \ldots, n,$$

$$J_k(i) = \min\Big[c(i) + c_r + \sum_{j=0}^{n} p_{ij}(u_1) J_{k+1}(j),\ c(i) + c_s + \sum_{j=0}^{n} p_{ij}(u_2) J_{k+1}(j)\Big],$$

$$k = 0, 1, \ldots, N-1$$

The two expressions in the minimization correspond to the two available decisions (fast and slow service).

1.3 DETERMINISTIC SYSTEMS AND THE SHORTEST PATH PROBLEM

The main objective of this text is the analysis of stochastic optimization problems and the ramifications of the presence of uncertainty. However,

deterministic problems arise in many important contexts, and the present and the next sections are devoted to explaining some of their distinguishing features.

We first note that deterministic problems can certainly be embedded within the framework of the basic problem simply by considering disturbance spaces D_k having a single element. However, in contrast with stochastic problems, *using feedback in deterministic problems results in no advantage in terms of cost reduction.* In other words, minimizing the cost functional over the class of admissible control laws $\{\mu_0, \ldots, \mu_{N-1}\}$ results in the same optimal cost as minimizing over the class of *sequences of control vectors* $\{u_0, \ldots, u_{N-1}\}$ with $u_k \in U_k(x_k)$ for all k. This is true simply because the cost achieved by an optimal control law $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ for a deterministic problem is also achieved by the control sequence

$$u_k^* = \mu_k^*(x_k^*), \qquad k = 0, \ldots, N-1,$$

where the states $x_0^*, \ldots, x_{N-1}^*$ are defined by

$$x_{k+1}^* = f_k(x_k^*, u_k^*), \qquad x_0^* = x_0, \qquad k = 0, 1, \ldots, N-1.$$

For this reason we may minimize the cost functional over sequences of controls, a task that may be achieved by variational deterministic optimal control algorithms such as steepest descent, conjugate gradient, and Newton's method. These algorithms, when applicable, are usually more efficient than DP. On the other hand, *DP has a wider scope of applicability since it can handle difficult constraint sets such as integer or discrete sets.* Furthermore, *DP leads to a globally optimal solution* as opposed to variational techniques, for which this cannot be guaranteed in general.

Consider now a deterministic problem where the state space S_k is a finite set for each k. Then at any state x_k a control u_k can be associated with a transition from the state x_k to the state $f_k(x_k, u_k)$. Thus a finite state deterministic problem can be equivalently represented by a graph such as the one of Figure 1.6, where the arcs correspond to transitions between states at successive stages and each arc has a cost associated with it. We have also added an artificial terminal node t. Each arc connecting a state x_N at stage N to the terminal node has cost $g_N(x_N)$. Control sequences correspond to paths originating at the initial state (node s at stage 0) and terminating at one of the nodes corresponding to the final stage N. If we view the cost of an arc as its length, we see that a *deterministic problem is equivalent to finding a shortest path from the initial node s of the graph to the terminal node t.* [A path is a sequence of arcs of the form (j_1, j_2), (j_2, j_3), $\ldots$, (j_{k-1}, j_k); its length is the sum of the length of its arcs.]

If we denote

$$c_{ij}^k = \text{cost of transition from state } i \in S_k \text{ to state } j \in S_{k+1},\ k = 0, 1, \ldots, N-1,$$

$$c_{it}^N = \text{terminal cost of state } i \in S_N,$$

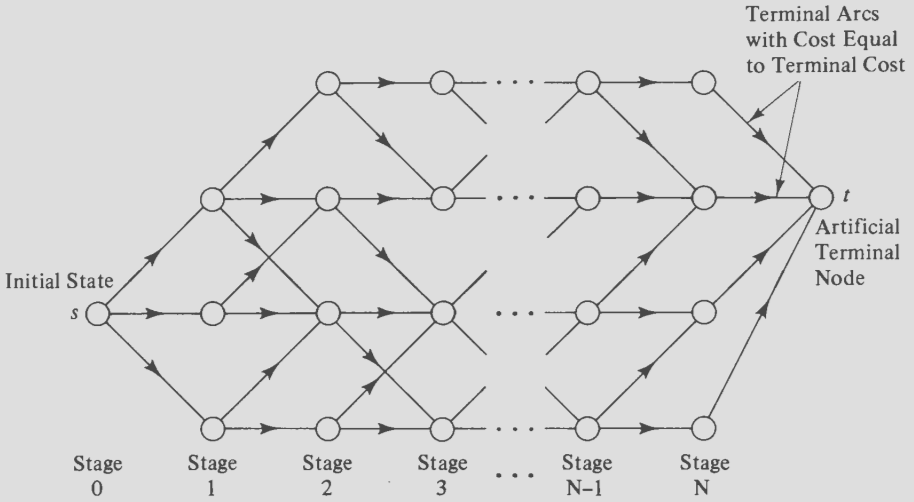


Figure 1.6 Transition graph for a deterministic finite state system. Nodes correspond to states. An arc with start and end nodes x_k and x_{k+1}, respectively, corresponds to a transition of the form $x_{k+1} = f_k(x_k, u_k)$. The length of this arc is equal to the cost of the corresponding transition $g_k(x_k, u_k)$. The problem is equivalent to finding a shortest path from the initial node s to the terminal node t.

the DP algorithm takes the form

$$J_N(i) = c_{it}^N, \qquad i \in S_N, \tag{1.9}$$

$$J_k(i) = \min_{j \in S_{k+1}} \{c_{ij}^k + J_{k+1}(j)\}, \qquad i \in S_k, \quad k = 0, 1, \ldots, N-1. \tag{1.10}$$

The optimal cost is $J_0(s)$ and equals the length of the shortest path from s to t.

The preceding algorithm proceeds *backward* in time. It is possible to derive an equivalent algorithm that proceeds *forward* in time by means of the following simple observation. An optimal path from s to t is also an optimal path from t to s in a "reverse" shortest path problem whereby the direction of each arc is reversed and its length is left unchanged. The DP algorithm corresponding to this "reverse" problem is

$$\tilde{J}_N(j) = c_{sj}^0, \qquad j \in S_1, \tag{1.11}$$

$$\tilde{J}_k(j) = \min_{i \in S_{N-k+1}} \{c_{ij}^{N-k} + \tilde{J}_{k+1}(i)\}, \qquad j \in S_{N-k+1},$$

$$k = 1, 2, \ldots, N-1, \tag{1.12}$$

and the optimal cost is

$$\tilde{J}_0(t) = \min_{i \in S_N} \{c_{it}^N + \tilde{J}_1(i)\}. \tag{1 13}$$

The backward and forward DP algorithms (1.9), (1.10) and (1.11) to (1.13), respectively, are equivalent in the sense that $J_0(s) = \tilde{J}_0(t)$, and an optimal control sequence (or shortest path) obtained from any one of the two is optimal for the original problem. We may view $\tilde{J}_k(j)$ in (1.12) as an *optimal cost-to-arrive* to state j from the initial state s. This should be contrasted with $J_k(i)$ in (1.10), which represents the optimal cost-to-go from state i to the terminal state t.

In conclusion, *a deterministic finite state problem is equivalent to a special type of shortest path problem and can be solved by either the ordinary (backward) DP algorithm or by an alternative forward DP algorithm.* It is also interesting to note that *any shortest path problem can be posed as a deterministic finite state DP problem,* as we now show.

Let $\{1, 2, \ldots, N, t\}$ be the set of nodes of a graph, and let c_{ij} be the cost of moving from node i to node j (or length of the arc joining i and j). Node t is a special node, which we call the *destination.* We allow the possibility $c_{ij} = \infty$ to account for the case where there is no arc joining nodes i and j. We want to find a shortest path from each node i to node t, that is, a sequence of moves that minimizes total cost to get to t from each of the nodes $1, 2, \ldots, N$. For the problem to have a solution, it is necessary to exclude the possibility that a sequence of moves that starts and ends at the same node (a cycle) has negative total length. Otherwise, it would be possible to decrease the length of some paths to arbitrarily small values simply by adding more and more negative-length cycles.

Since negative-length cycles have been excluded by assumption, it is clear that an optimal path need not take more than N moves, so we may limit the number of moves to N. We formulate the problem as one where *we require exactly N moves but allow degenerate moves from a node i to itself with cost* $c_{ii} = 0$. We denote for $i = 1, \ldots, N$, $k = 0, 1, \ldots, N - 1$,

$J_{N-1}(i)$ = optimal cost for getting from i to t in one move,

$J_k(i)$ = optimal cost for getting from i to t in $(N - k)$ moves.

Then the cost of the optimal path from i to t is $J_0(i)$. It is possible to formulate this problem within the framework of the basic problem and subsequently apply the DP algorithm. For simplicity, however, we write directly the DP equation, which takes the intuitively clear form

optimal cost from i to t in $(N - k)$ moves

$$= \min_{j=1,\ldots,N} \{c_{ij} + \text{optimal cost from } j \text{ to } t \text{ in } (N - k - 1) \text{ moves}\},$$

or

$$J_k(i) = \min_{j=1,\ldots,N} \{c_{ij} + J_{k+1}(j)\}, \qquad k = 0, 1, \ldots, N - 2,$$

with

$$J_{N-1}(i) = c_{it}, \qquad i = 1, 2, \ldots, N.$$

The optimal policy when at node i after k moves is to move to node j^*, where j^* minimizes over all $j = 1, \ldots, N$ the expression in braces. Note that a degenerate move from i to i is not excluded. If the optimal path obtained from the algorithm contains such degenerate moves, this simply means that its duration is less than N moves.

To demonstrate the algorithm, consider the problem shown in Figure 1.7a, where the costs c_{ij}, $i \neq j$ (we assume $c_{ij} = c_{ji}$), are shown along the connecting line segments. Figure 1.7b shows the cost-to-go $J_k(i)$ at each node i and index k together with the optimal path. The optimal paths are

$$1 \to 5, \quad 2 \to 3 \to 4 \to 5, \quad 3 \to 4 \to 5, \quad 4 \to 5.$$

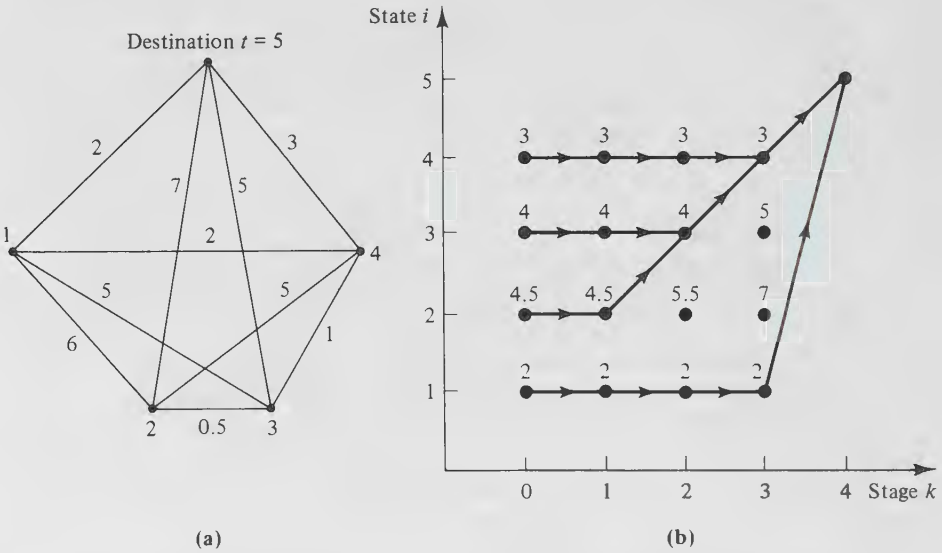


Figure 1.7 (a) Shortest path problem data. The destination is 5. Arc lengths are equal in both directions and are shown along the line segments connecting nodes. (b) Costs-to-go generated by the DP algorithm. The number along stage k and state i is $J_k(i)$. Arrows indicate the optimal moves at each stage and node.

1.4 SHORTEST PATH APPLICATIONS IN CRITICAL PATH ANALYSIS, CODING THEORY, AND FORWARD SEARCH

The shortest path problem appears in many diverse contexts. We provide some examples.

Critical Path Analysis

Consider the planning of a project involving several activities, some of which must be completed before others can begin. The duration of each

activity is known in advance. We want to find the time required to complete the project, as well as the *critical* activities, those that even if slightly delayed will result in a corresponding delay of completion of the overall project.

The problem can be represented by a directed graph with nodes 1, . . ., N such as the one shown in Figure 1.8 (also called an *activity network*). Here nodes represent completion of some phase of the project. An arc (i, j) represents an activity that starts once phase i is completed and has known duration t_{ij}. A phase (node) j is completed when all activities or arcs (i, j) that are incoming to j are completed. The special nodes 1 and N represent the start and end of the project. Naturally, node 1 (N) has no incoming (outgoing) arcs.

An important characteristic of an activity network is that it is *acyclic;* that is, it has no directed cycles [sequences of directed arcs of the form $(i, j_1), (j_1, j_2), \ldots, (j_k, i)$]. This is inherent in the problem formulation and the interpretation of nodes as phase completions.

Consider now the time T required to complete all phases of the project and hence the project itself. For any directed path $p = \{(1, j_1), (j_1, j_2), \ldots, (j_{k-1}, j_k)\}$ from node 1 to node j_k, let D_p be the duration of the path defined as the sum of durations of its activities; that is,

$$D_p = t_{ij_1} + t_{j_1j_2} + \cdots + t_{j_{k-1}j_k}.$$

So D_p is the total duration of the sequence of activities $(1, j_1), \ldots, (j_{k-1}, j_k)$ if each could be started immediately after the previous ended. Clearly, D_p cannot exceed the total project duration time; that is,

$$D_p \leq T, \qquad \text{all paths } p.$$

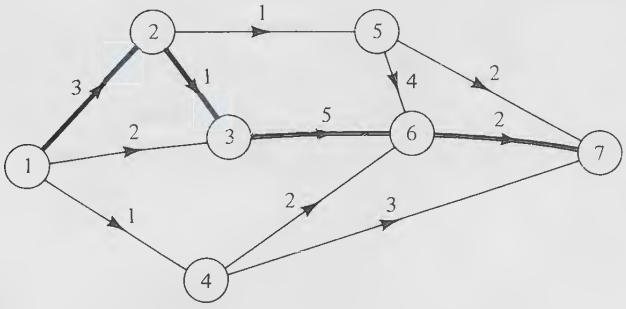


Figure 1.8 Graph of an activity network. Nodes represent completion of some phase of the project. Arcs represent activities and are labeled by the duration. A phase is completed if all activities associated with incoming arcs at the corresponding node are completed. The project is completed when all phases are completed. The project duration time is the length of the longest path from node 1 to node 7.

We claim that

$$T = \max_p D_p,$$

and therefore finding T may be viewed as a problem of finding the *longest* path from node 1 to node N when the length of each arc (i, j) is t_{ij}. Because the graph is acyclic, this problem may also be viewed as a shortest path problem with the length of each arc (i, j) being $-t_{ij}$.

The easiest way to show this is by deriving the corresponding DP algorithm. Let N_k, $k = 1, 2, \ldots$, be the set of phases

$$N_k = \{i \mid \text{the maximum number of arcs contained in paths from 1 to } i \text{ is exactly } k\}$$

with $N_0 = \{1\}$. For each phase i, let

$$T_i: \quad \text{required time to complete } i.$$

Then we have

$$T_i = \max_{(j,i)}\{t_{ji} + T_j \mid j \in N_0 \cup \cdots \cup N_{k-1}\}, \qquad i \in N_k,$$

and a little thought reveals that T_i equals the maximum D_p over all paths p from 1 to i For $i = N$, we obtain $T = \max_p D_p$.

For the activity network of Figure 1.8, we have

$$N_0 = \{1\}, \quad N_1 = \{2, 4\}, \quad N_2 = \{3, 5\} \quad N_3 = \{6\}, \quad N_4 = \{7\}.$$

A calculation using the preceding formula yields

$$T_1 = 0, \quad T_2 = 3, \quad T_4 = 1, \quad T_3 = 4, \quad T_5 = 4, \quad T_6 = 9, \quad T_7 = 11,$$

and the critical (i.e., longest) path is $1 \to 2 \to 3 \to 6 \to 7$. Any delay in the completion of the critical activities (1, 2), (2, 3), (3, 6), (6, 7) will proportionately delay the completion of the overall project.

Convolutional Coding and the Viterbi Decoder

When binary data are transmitted over a noisy communication channel, it is often essential to use coding as a means of enhancing reliability of communication. A very common type of coding method, called *convolutional coding*, converts a source-generated binary data sequence

$$\{w_1, w_2, \ldots\}, \qquad w_k \in \{0, 1\}, \quad k = 1, 2, \ldots,$$

into a coded sequence

$$\{y_1, y_2, \ldots\},$$

where each y_k, $k = 1, 2, \ldots$, is an n-dimensional vector with binary coordinates (called codeword)

$$y_k = \begin{bmatrix} y_k^1 \\ \cdots \\ y_k^n \end{bmatrix}, \qquad y_k^i \in \{0, 1\}, \quad i = 1, \ldots, n.$$

The vectors y_k are related to w_k via equations of the form

$$y_k = Cx_{k-1} + dw_k, \qquad k = 1, 2, \ldots \tag{1.14}$$

$$x_k = Ax_{k-1} + bw_k, \qquad k = 1, 2, \ldots, \tag{1.15}$$

$$x_0: \quad \text{given},$$

where x_k is an m-dimensional vector with binary coordinates (called state) and C, d, A, and b are $n \times m$, $n \times 1$, $m \times m$, and $m \times 1$ matrices, respectively, with binary coordinates. The products and the sums involved in the expressions $Cx_{k-1} + dw_k$ and $Ax_{k-1} + bd$, are calculated using modulo 2 arithmetic.

As an example, let $m = 2$, $n = 3$, and

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \qquad A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix},$$

$$d = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then the evolution of the system (1.14) to (1.15) can be represented by the transition diagram (called a *trellis*) shown in Figure 1.9. From this diagram and the initial x_0, it is possible to generate the codeword sequence $\{y_1, y_2, \ldots\}$ corresponding to a data sequence $\{w_1, w_2, \ldots\}$. For example, when the

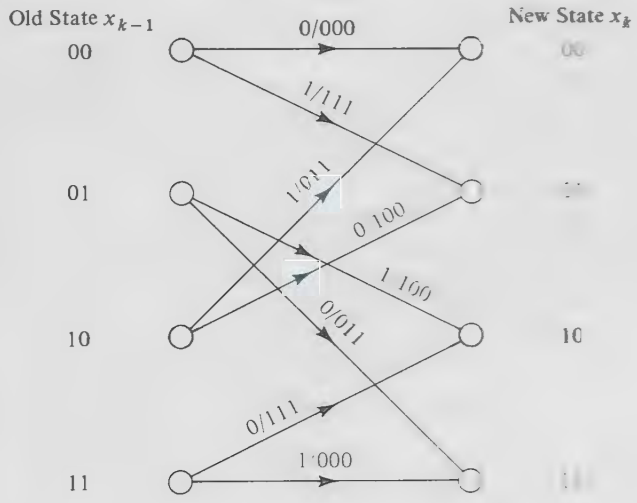


Figure 1.9 State transition diagram from x_{k-1} to x_k. The binary number pair on each arc is the data/codeword pair w_k/y_k for the corresponding transition. So, for example, when $x_{k-1} = 01$, a zero data bit ($w_k = 0$) effects a transition to $x_k = 11$ and generates the codeword 011.

initial state is $x_0 = 00$, the data sequence

$$\{w_1, w_2, w_3, w_4\} = \{1, 0, 0, 1\}$$

generates the state sequence

$$\{x_0, x_1, x_2, x_3, x_4\} = \{00, 01, 11, 10, 00\},$$

and the codeword sequence

$$\{y_1, y_2, y_3, y_4\} = \{111, 011, 111, 011\}.$$

Assume now that the characteristics of the noisy transmission channel are such that a codeword y is actually received as z with known probability $p(z \mid y)$, where z is any n-bit binary number. We denote

$$Z_N = \{z_1, z_2, \ldots, z_N\}$$

the sequence received when the transmitted sequence is

$$Y_N = \{y_1, y_2, \ldots, y_N\}.$$

We assume independent errors so that

$$p(Z_N \mid Y_N) = \prod_{k=1}^{N} p(z_k \mid y_k). \tag{1.16}$$

A *maximum likelihood decoder* converts a received sequence Z_N into a sequence

$$\hat{Y}_N = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N\}$$

such that

$$p(Z_N \mid \hat{Y}_N) = \max_{Y_N} p(Z_N \mid Y_N).$$

The constraint on Y_N is that it must be a feasible codeword sequence (i.e., it must correspond to some initial state and data sequence). Given $\hat{Y}_N$, one can then construct a corresponding data sequence $\{\hat{w}_1, \ldots, \hat{w}_N\}$ that is accepted as the decoded data.

Viterbi developed a shortest path scheme that implements the maximum likelihood decoder. Using (1.16), we see that the problem of maximizing $p(Z_N \mid Y_N)$ is equivalent to the problem

$$\begin{aligned} &\text{minimize} \sum_{k\ 1}^{N} - \ln[p(z_k \mid y_k)] \\ &\text{over all binary sequences } \{y_1, y_2, \ldots, y_N\} \end{aligned} \tag{1.17}$$

for a known received sequence $\{z_1, z_2, \ldots, z_N\}$. To see that this is a shortest path problem, note that, given z_k, *we can assign to each arc on the state transition diagram the length* $-\ln[p(z_k \mid y_k)]$, *where* y_k *is the codeword associated with the arc*. Next we construct a graph by concatenating N state transition diagrams and appending dummy nodes s and t on the left and right side of the graph connected with zero-length arcs to the states x_0 and x_{N-1}, respectively (see Figure 1.10). The solution to problem (1.17)

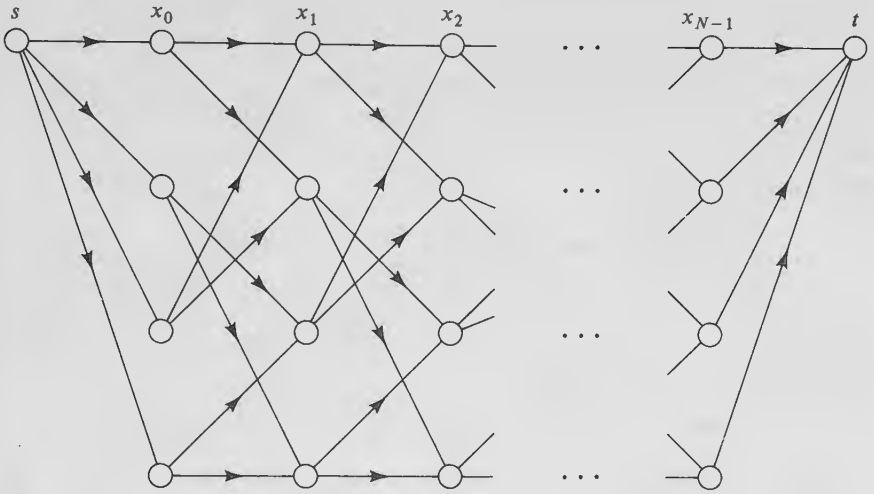


Figure 1.10 Maximum likelihood decoding viewed as a problem of finding a shortest path from s to t. Length of arcs from s to states x_0 and from states x_{N-1} to t is zero. Length of an arc from a state x_{k-1} to x_k is $-\ln p(z_k \mid y_k)$, where z_k is the received codeword and y_k is the codeword associated with the arc.

is obtained by constructing a shortest path from s to t and finding the associated sequence $\{\hat{y}_1, \ldots, \hat{y}_N\}$. From the shortest path and the trellis diagram, we can then obtain the decoded data sequence $\{\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_N\}$.

In practice, the shortest path is most conveniently constructed by calculating the shortest distance from s to each node on-line as soon as the corresponding codeword is received. There are a number of practical schemes for decoding a portion of the data sequence prior to receiving the entire codeword sequence Z_N. (This is useful if Z_N is a long sequence.) For example, one can check rather easily whether for some k all shortest paths from s to states x_k pass through a single node in the subgraph of states $x_0, \ldots, x_{k-1}$. If so, it can be seen that the shortest path from s to that node will not be affected by reception of additional codewords (the principle of optimality), and therefore the corresponding data subsequence can be safely decoded and delivered to its destination.

Forward Search

In some shortest path problems the number of nodes is extremely large. As a result, storing these nodes in a computer's memory can be very difficult. Indeed, the nature of some shortest path problems is such that the solution becomes very simple once the nodes of the underlying graph are enumerated, and the real issue is how to solve the problem while avoiding a complete enumeration of all nodes. In such cases it is frequently

possible to save both in memory and in computation by means of a forward search for a shortest path from an origin node toward a destination node. The techniques for doing this have partly originated in artificial intelligence and are typically used in computer progams that solve puzzles or play games such as chess (see Section 4.3). Let us provide some examples.

Example 1

The Four Queens Problem. Four queens must be placed on a 4×4 portion of a chessboard so that no queen can attack another. In other words, the placement must be such that every row, column, or diagonal of the 4×4 board contains at most one queen. Equivalently, we can view the problem as a sequence of problems; first, placing a queen in one of the first two squares in the top row, then placing another queen in the second row so that it is not attacked by the first, and similarly placing the third and fourth queens. (It is sufficient to consider only the first two squares of the top row since the other two squares lead to symmetric positions.) We can associate positions with nodes of an acyclic graph where the root node s corresponds to the position with no queens and the terminal nodes correspond to the dead-end positions where no additional queens can be placed without some queen attacking another. Let us connect each terminal position with an artificial node t by means of an arc. Let us also assign to all arcs length zero except for the artificial arcs connecting terminal positions with less than four queens with the artificial node t. These latter arcs are assigned the length $+\infty$ (see Figure 1.11) to express the fact that they correspond to dead-end positions that cannot lead to a solution. Then the four queens problem reduces to finding a shortest path from node s to node t. Note that once the nodes of the graph are enumerated the problem is essentially solved. Here the number of nodes is small. However, we can think of similar problems with much larger memory requirements. For example, there is an eight queens problem where the board is 8×8 instead of 4×4.

Example 2

The Traveling Salesman Problem. An important model for scheduling a sequence of operations is the classical traveling salesman problem. Here we are given N cities and the mileage between each pair of cities, and we wish to find a minimum-mileage trip that visits each of the cities exactly once. To convert this problem to a shortest path problem, we associate a node with every sequence of n distinct cities, where $n = 1, 2, \ldots, N$. The construction and arc lengths of the corresponding graph are explained by means of an example in Figure 1.12. The origin node s consists of city A, taken as the start. A sequence of n cities ($n < N$) yields a sequence of $(n + 1)$ cities by adding a new city. Two such sequences are connected by an arc with length equal to the mileage between the last two of the $n + 1$ cities. Each sequence of N cities is connected to an artificial terminal node t with an arc having length equal to the distance from the last city of the sequence to the starting city A. Note that the number of nodes grows exponentially with the number of cities, so we would like to have algorithms that do not require the enumeration and/or storage of these nodes

In the shortest path problem that we will consider there is a single node s with no incoming arcs, called the *origin*, and a single node t with no outgoing arcs, called the *destination*. We assume that every arc (i, j)

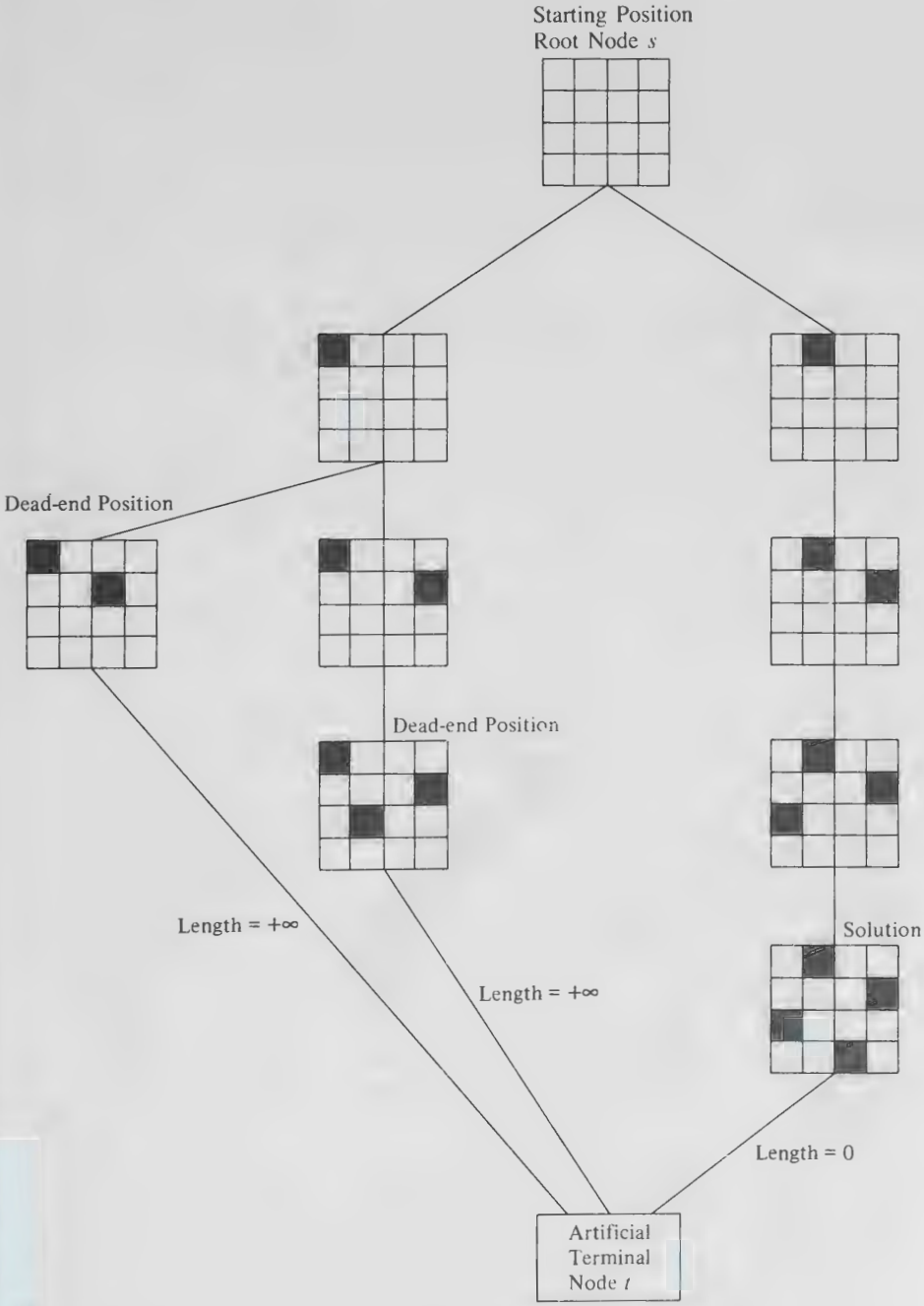
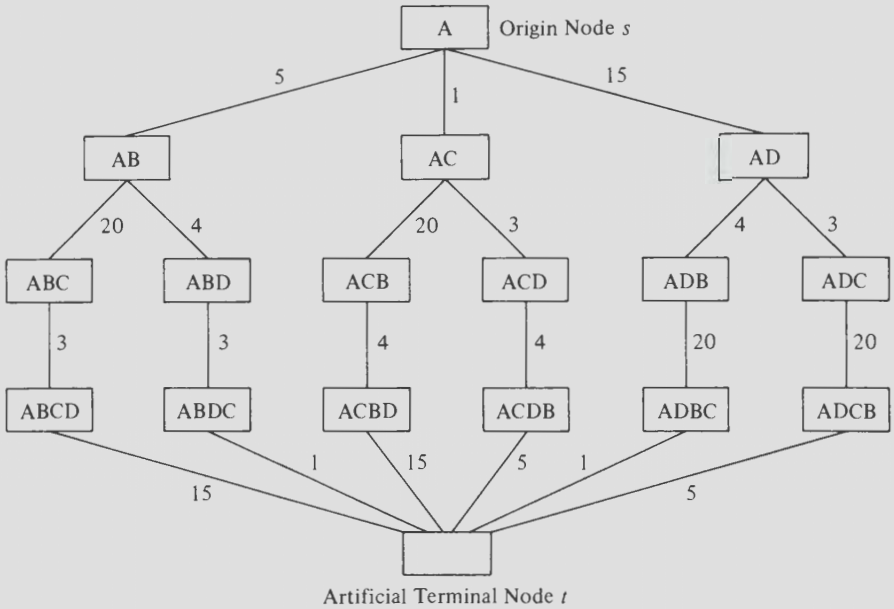


Figure 1.11 Shortest path formulation of the four queens problem. Symmetric positions resulting from placing a queen in one of the rightmost squares in the top row have been ignored. Squares containing a queen have been darkened. All arcs have length zero except for those connecting dead-end positions to the artificial terminal node.



	A	B	C	D
A		5	1	15
B	5		20	4
C	1	20		3
D	15	4	3	

Table of Mileage between Cities

Figure 1.12 Example of shortest path formulation of the traveling salesman problem. The distance between the four cities A, B, C, and D are shown in the table. The arc lengths are shown next to the arcs.

has a length a_{ij} which is nonnegative or $+\infty$, and we wish to find a shortest path from origin to destination. We assume that there exists a shortest path with finite length. The following algorithm is a general method for solving the problem. In it we make use of two lists of nodes called OPEN and CLOSED. The list OPEN contains nodes that are currently active in the sense that they are candidates for further examination by the algorithm. The list CLOSED contains nodes that have been examined by the algorithm and are not currently candidates for further consideration. Using CLOSED is not essential for the algorithm, but results in some conceptual simplification. Initially, OPEN contains

just the origin node s and CLOSED is empty. The algorithm maintains an upper bound of the shortest distance from origin to destination called UPPER and initially equal to $+\infty$. The algorithm also maintains for each node i an upper bound d_i of its shortest distance from the origin. Initially, $d_s = 0$ and $d_i = +\infty$ for all other nodes i. A node j is called a *son* of node i if there is an arc (i, j) connecting i with j. The steps of the algorithm are as follows:

Step 1 Remove a node i from the top of OPEN and place it in CLOSED. For each son j of i, go to step 1a if $j \neq t$, and go to step 1b if $j = t$.

Step 1a ($j \neq t$) If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_i + a_{ij}$, give j the label i, place j at the top of OPEN, and remove j from CLOSED if it belongs there. (*Note:* The label is needed in order to trace the shortest path to the origin after the algorithm terminates.)

Step 1b ($j = t$): If $d_i + a_{it} < \text{UPPER}$, set $\text{UPPER} = d_i + a_{it}$, and mark node i as lying on the best path found so far from s to t.

Step 2 If OPEN is empty, terminate; else go to step 1.

It can be seen that, throughout the algorithm, d_j is either $+\infty$ (if node j has not yet entered the OPEN list), or else it is the length of a path from s to j consisting of nodes that have entered the OPEN list at least once. Furthermore, UPPER is either $+\infty$, or else it is the length of a path from s to t, and consequently it is an overestimate of the shortest distance from s to t. The idea in the algorithm is that when a shorter path from s to j is discovered than those considered earlier ($d_i + a_{ij} < d_j$ in step 1a), the value of d_j is accordingly reduced, and node j enters the OPEN list so that paths passing through j and reaching the sons of j can be taken into account. It makes sense to do so, however, only when the path considered has a chance of leading to a path from s to t with length smaller than the overestimate UPPER of the shortest distance from s to t. In view of the nonnegativity of the arc lengths, this is not possible if the path length $d_i + a_{ij}$ is not smaller than UPPER. This provides the rationale for entering j into OPEN in step 1a only if $d_i + a_{ij}$ is less than UPPER.

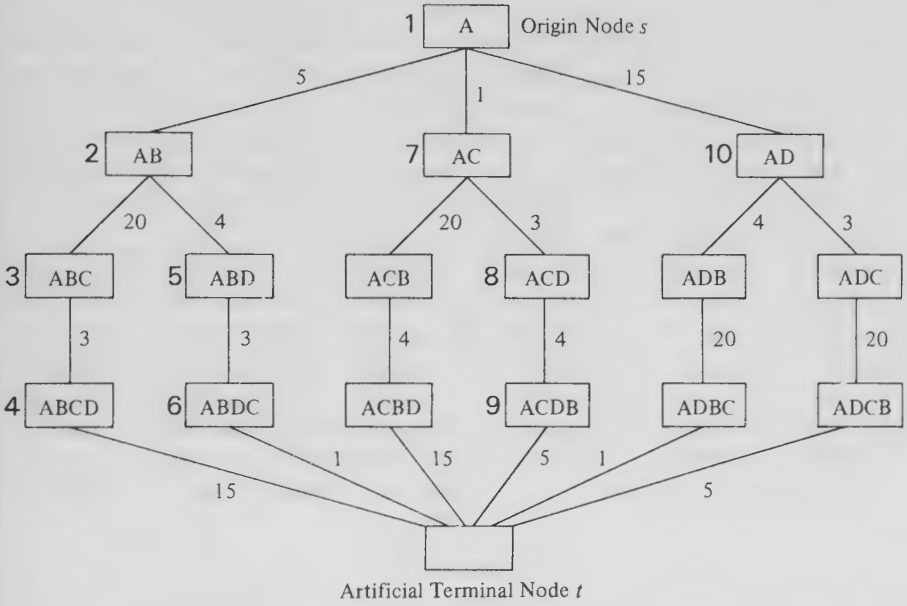
Tracing the algorithm, we see that it will first examine node s (the only node initially in OPEN), place s (permanently) in CLOSED, and assuming t is not a son of s, it will place all the sons j of s in OPEN after setting $d_j = a_{sj}$. If t is a son of s, then UPPER will be set to a_{st} in step 1b, and the sons of s examined after t will be placed in OPEN only if $a_{sj} < a_{st}$; indeed, this should be so since if $a_{sj} \geq a_{st}$ node j cannot lie on a shorter path from s to t than the direct path consisting of arc (s, t). The algorithm will subsequently take the last son $j \neq t$ of s from the top of OPEN, place

it in CLOSED, and place those of its sons $j \neq t$ that satisfy the criterion of step 1a in OPEN, etc. When the algorithm terminates, we claim that a shortest path can be obtained by using the node last marked in step 1b as lying on the best path. By tracing labels starting from that node we can proceed backward and construct a shortest path to the origin node. Fig. 1.13 illustrates the use of the algorithm to solve the traveling salesman problem of Fig. 1.12.

To verify that a path obtained as just described is shortest, we reason as follows. We first argue by contradiction that the algorithm will terminate. Indeed, if this is not so, some node j will enter the OPEN list infinitely often, which means that d_j will be decreased infinitely often, each time obtaining a corresponding shorter path from s to j. This is not possible since, in view of the nonnegative arc assumption, the number of distinct lengths of paths from s to j is finite. Therefore, the algorithm will terminate. We next show that the value of UPPER upon termination must equal the shortest distance d^* from s to t. Indeed, let $(s, j_1, j_2, \ldots, j_k, t)$ be a shortest path from s to t. Then each path $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, is a shortest path from s to j_m, respectively. If the value of UPPER is larger than d^* at termination, the same must be true throughout the algorithm, and therefore UPPER will also be larger than the length of all the paths $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, throughout the algorithm. It follows that node j_k will never enter the OPEN list with d_{j_k} equal to the shortest distance from s to j_k, since in this case UPPER would be set to d^* in step 1b immediately following the next time node j_k is examined by the algorithm in step 1. Similarly, this means that node j_{k-1} will never enter the OPEN list with $d_{j_{k-1}}$ equal to the shortest distance from s to j_{k-1}. Proceeding backward, we conclude that j_1 never enters the OPEN list with d_{j_1} equal to the shortest distance from s to j_1 (which is equal to the length of the arc (s, j_1)). This happens, however, at the first iteration of the algorithm as discussed earlier, so we have reached a contradiction. It follows that UPPER will equal at termination the shortest distance from s to t. It is seen that the path constructed by tracing labels backward from t to s has length equal to UPPER, so it is a shortest path from s to t.

There are two attractive aspects to this algorithm. The first is a potential saving in computation in that nodes j for which $d_i + a_{ij} \geq$ UPPER in step 1a need not enter OPEN and be examined later. Furthermore, if we know a lower bound to the shortest distance, we can terminate the computation once UPPER reaches that bound either exactly or within an acceptable tolerance $\varepsilon > 0$. (This feature is useful, for example, in the four queens problem, where the shortest distance is known to be zero or infinity. Then the algorithm will terminate once a solution is found.)

The second attractive aspect of the algorithm is a potential saving in memory storage requirements. This is most evident in graphs such as those in Figures 1.11 and 1.12 for which there is a unique directed path from the



(a)

Iteration No.	List OPEN	Node Entering CLOSED	UPPER
0	1	–	$+\infty$
1	2, 7, 10	1	$+\infty$
2	3, 5, 7, 10	2	$+\infty$
3	4, 5, 7, 10	3	$+\infty$
4	5, 7, 10	4	43
5	6, 7, 10	5	43
6	7, 10	6	13
7	8, 10	7	13
8	9, 10	8	13
9	10	9	13
10	Empty	10	13

(b)

Figure 1.13 The algorithm applied to the traveling salesman problem of Figure 1.12. The optimal solution ABDC is found after examining nodes 1 through 10 in that order. The table shows the successive contents of the OPEN list.

origin node to every other node. Then, in view of our convention of placing nodes at and removing nodes from the top of OPEN, the search proceeds in depth-first fashion, as shown in Figure 1.14. As a result, large portions of CLOSED can be purged from memory, as shown in Figure 1.15. The basis for this is that once all sons of a node enter the CLOSED list then all paths passing through that node have been generated and evaluated. Therefore, it is sufficient to store only the best path found so far and purge all other information relating to such a node.

There are a number of variations of the algorithm just given. The preceding proof of validity of the algorithm does not depend on removing a node from the top of OPEN in step 1 or placing a node at the top of OPEN in step 1a. This allows a great deal of freedom on how the algorithm is operated. An important case is when the node i selected in step 1 is not the node that happens to be at the top of OPEN, but rather the one in OPEN for which d_i is *minimum*. This is accordingly known as *best-first search* and is equivalent for the problem considered here to Dijkstra's

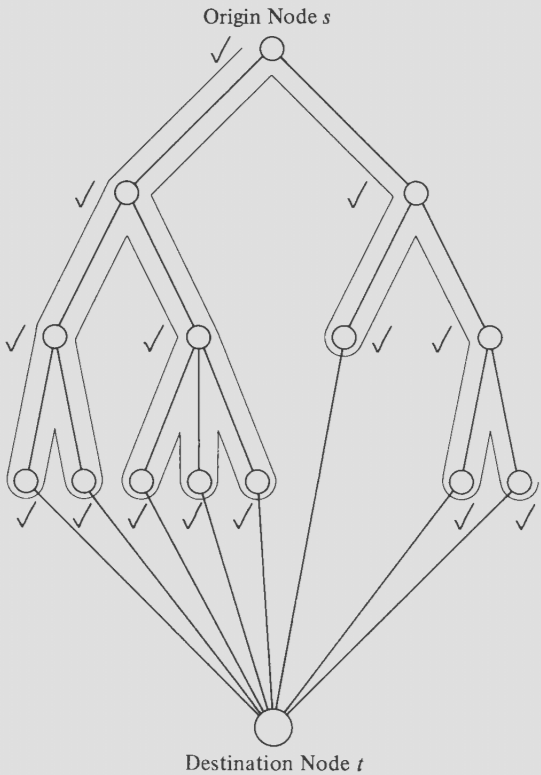


Figure 1.14 Searching a tree in depth-first fashion. The checkmarks show the order in which nodes enter the CLOSED list.

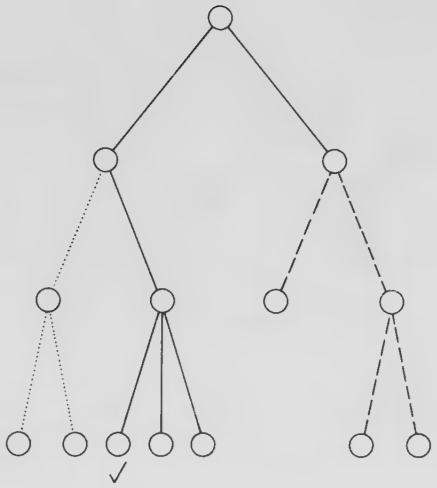


Figure 1.15 Memory requirements of depth-first search for the graph of Figure 1.14. At the time the node marked by the checkmark enters the CLOSED list, only the solid-line portion of the tree is needed in memory. The dotted-line portion has been generated and purged from memory based on the rule that it is unnecessary to store a node with all successors in CLOSED. The broken-line portion of the tree has not yet been generated.

algorithm (see [P2] and Problem 27). Another possibility is *to place in step 1a the node j at the top of OPEN if j currently belongs to CLOSED, to the bottom of OPEN if j does not belong to CLOSED or OPEN, and to leave j in its current position in OPEN if it belongs to OPEN.* This algorithm was suggested by Pape [P4] and turns out to be very effective for important classes of problems [D6].

We mentioned earlier that the key idea of the algorithm is to save computation by foregoing the examination of nodes j that cannot lie on a shortest path. This is based on the test $d_i + a_{ij} <$ UPPER that node j must pass before it can be placed in the OPEN list in step 1a. We can strengthen this test if we can find a *positive underestimate* h_j of the shortest distance of node j to the destination. Such an estimate can be obtained from special knowledge about the problem at hand. We may speed up the computation substantially by placing a node j in OPEN in step 1a when $d_i + a_{ij} + h_j <$ UPPER (instead of $d_i + a_{ij} <$ UPPER). In this way, fewer nodes will potentially be placed in OPEN before termination. Using the fact that h_j is an underestimate of the true shortest distance from j to the destination, the argument given earlier shows that the algorithm will terminate with a correct shortest path.

The idea just described is one way to sharpen the test $d_i + a_{ij} <$ UPPER for admission of node j into the OPEN list. An alternative idea is to try to reduce the value of UPPER by obtaining for the node j in step 1a an *overestimate* h_j of the shortest distance from j to the destination. Then if $d_j + h_j <$ UPPER after step 1a, we can reduce UPPER to $d_j + h_j$, thereby making the test for future admissibility into OPEN more stringent. This idea is used in some versions of the branch-and-bound algorithm, one of which we now briefly describe (see also [P2] and [P9] for further discussion).

Example 3

Branch-and-Bound Algorithm. Consider a problem of minimizing a cost function $f(x)$ over a *finite* set of feasible solutions X. The branch-and-bound algorithm uses an acyclic graph with nodes that correspond on a one-to-one basis with a collection $\mathcal{N}$ of subsets of X. We require the following:

1. $X \in \mathcal{N}$ (i.e., the set of all solutions is a node).
2. If x is a solution, then $\{x\} \in \mathcal{N}$ (i.e., all solutions viewed as singleton sets are nodes).
3. If $Y \in \mathcal{N}$ contains more than one solution $x \in X$, then there exist $Y_1, \ldots, Y_n \in \mathcal{N}$ such that $Y_i \neq Y$ for all i and
$$\bigcup_{i=1}^{n} Y_i = Y.$$
Y is called the *parent* of $Y_1, \ldots, Y_n$, and $Y_1, \ldots, Y_n$ are called the *sons* of Y.
4. Each node other than X has at least one parent.

It is clear that $\mathcal{N}$ defines an acyclic graph with root node X and terminal nodes $\{x\}$, $x \in X$ (see Figure 1.16). If Y_i is a son of Y, we assume that there is an arc connecting Y and Y_i. Suppose that for every node Y there is an algorithm that calculates upper and lower bounds $\underline{f}_Y$ and $\bar{f}_Y$ for the minimum cost over Y, that is:

$$\underline{f}_Y \leq \min_{x \in Y} f(x) \leq \bar{f}_Y.$$

Assume further that the upper and lower bounds are exact for a singleton solution node,

$$\underline{f}_{\{x\}} = f(x) = \bar{f}_{\{x\}}, \qquad \text{for all } x \in X.$$

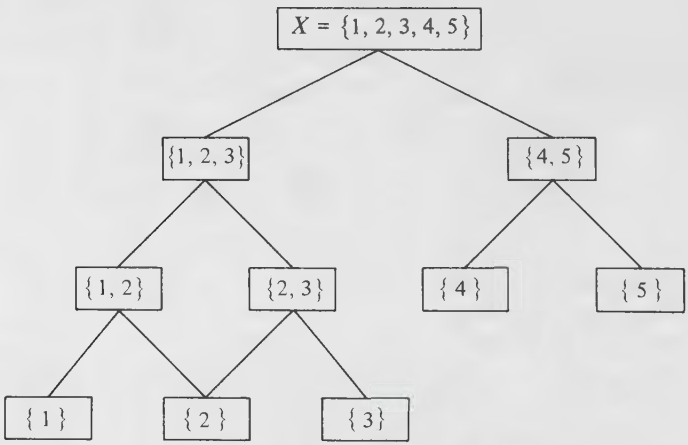


Figure 1.16 A tree corresponding to a branch-and-bound algorithm. Each node (subset) except those consisting of a single solution is partitioned into several other nodes (subsets).

Define now the length of an arc involving a parent Y and a son Y_i to be the lower bound difference

$$\underline{f}_{Y_i} - \underline{f}_Y.$$

Then evidently for every node Y the lower bound $\underline{f}_Y$ is $\underline{f}_X$ plus the length of any path from the origin node X to Y. Because of our assumption $\underline{f}_{\{x\}} = f(x)$ for all feasible solutions $x \in X$, it is clear that finding a shortest path from the origin node to one of the singleton nodes is equivalent to minimizing $f(x)$ over $x \in X$.

Consider now a variation of the shortest path algorithm discussed earlier where in addition we use our knowledge of the upper bounds $\bar{f}_Y$ to reduce the value of UPPER. Initially, OPEN contains just X, and UPPER equals $\bar{f}_X$.

Step 1 Remove a node Y from OPEN. For each son Y_j of Y, execute Step 2.

Step 2 If $\underline{f}_{Y_j} <$ UPPER, then place Y_j in OPEN. If in addition $\bar{f}_{Y_j} <$ UPPER, then set UPPER $= \bar{f}_{Y_j}$, and if Y_j consists of a single solution, mark that solution as being the best solution found so far.

Step 3 If OPEN is nonempty, go to step 1. Otherwise, terminate; the best solution found so far is optimal.

An alternative termination step 3 for the preceding algorithm is to set a tolerance $\epsilon > 0$ and check whether UPPER and the minimum lower bound $\underline{f}_Y$ over all sets Y in the OPEN list differ by less than ϵ. If so, the algorithm is terminated, and some set in OPEN must contain a solution within ϵ of being optimal. There are a number of other variations of the algorithm. For example, if the upper bound $\bar{f}_Y$ at a node is actually the cost $f(x)$ of some element $x \in Y$, then this element can be taken as the best solution found so far whenever $\bar{f}_Y <$ UPPER in step 2. Other variations relate to the method of selecting a node from OPEN in step 1. For example, two strategies of the best-first type are to select the node with minimal lower or upper bound. In closing, we note that applying branch and bound effectively requires the creative use of knowledge of the particular problem at hand. In particular, it is important to have algorithms for generating as sharp as practically possible upper and lower bounds at each node, since then fewer nodes will be admitted into OPEN, with attendant computational savings.

1.5 TIME LAGS, CORRELATED DISTURBANCES, AND FORECASTS

This section deals with situations where some of the assumptions in the basic problem formulation are not satisfied. We shall consider the case where there are time lags in the system equation, the case where the disturbances w_k are correlated, and the case where at time k a forecast on the future uncertainties w_k, w_{k+1}, . . . becomes available, thus updating the corresponding probability distributions. The situation where the system evolution may terminate prior to the final time either due to a random event

or due to an action of the decision maker is covered in the problems. Generally, in all these cases it is possible to reformulate the problem into the framework of the basic problem by using the device of *state augmentation*. The (unavoidable) price paid, however, is an increase in complexity of the reformulated problem.

Time Lags

For simplicity, assume that there is at most a single period time lag in the state and control, that is, assume a system equation of the form

$$\begin{aligned} x_{k+1} &= f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k), \qquad k = 1, 2, \ldots, N-1, \\ x_1 &= f_0(x_0, u_0, w_0). \end{aligned} \tag{1.18}$$

Time lags of more than one period can be handled by a straightforward extension.

Now if we introduce additional state variables y_k and s_k and make the identifications $y_k = x_{k-1}$, $s_k = u_{k-1}$, the system equation (1.18) yields, for $k = 1, 2, \ldots, N-1$,

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ s_{k+1} \end{bmatrix} = \begin{bmatrix} f_k(x_k, y_k, u_k, s_k, w_k) \\ x_k \\ u_k \end{bmatrix} \tag{1.19}$$

By defining $\bar{x}_k = (x_k, y_k, s_k)$ as the new state, we have

$$\bar{x}_{k+1} = \bar{f}_k(\bar{x}_k, u_k, w_k), \tag{1.20}$$

where the system function $\bar{f}_k$ is defined in an obvious manner from (1.19). By using (1.20) as the system equation and by making a suitable reformulation of the cost functional, the problem is reduced to the basic problem without time lags. Naturally, the control law $\{\mu_0, \ldots, \mu_{N-1}\}$ that is sought will consist of functions μ_k of the new state $\bar{x}_k$, or equivalently μ_k *will be a function of the present state* x_k *as well as past state* x_{k-1} *and control* u_{k-1}. The DP algorithm (in terms of the variables of the original problem) is

$$J_N(x_N) = g_N(x_N),$$

$$J_{N-1}(x_{N-1}, x_{N-2}, u_{N-2}) = \min_{u_{N-1} \in U_{N-1}(x_{N-1})} E_{w_{N-1}} \{g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) + J_N[f_{N-1}(x_{N-1}, x_{N-2}, u_{N-1}, u_{N-2}, w_{N-1})]\},$$

$$J_k(x_k, x_{k-1}, u_{k-1}) = \min_{u_k \in U_k(x_k)} E_{w_k} \{g_k(x_k, u_k, w_k) + J_{k+1}[f_k(x_k, x_{k-1}, u_k, u_{k-1}, w_k), x_k, u_k]\}, \qquad k = 1, \ldots, N-2,$$

$$J_0(x_0) = \min_{u_0 \in U_0(x_0)} E_{w_0} \{g_0(x_0, u_0, w_0) + J_1[f_0(x_0, u_0, w_0), x_0, u_0]\}.$$

We note that similar reformulations are possible when time lags appear in the cost functional, for example, in the case where the expression to be minimized is of the form

$$E\left\{g_N(x_N, x_{N-1}) + \sum_{k=0}^{N-1} g_k(x_k, x_{k-1}, u_k, w_k)\right\}.$$

The extreme case of time lags in the cost functional is when it has the nonadditive form

$$E\{g_N(x_N, x_{N-1}, \ldots, x_0, u_{N-1}, \ldots, u_0, w_{N-1}, \ldots, w_0)\}.$$

Then, to reduce the problem to the form of the basic problem, the augmented state $\tilde{x}_k$ at time k must include

$$(x_k, x_{k-1}, \ldots, x_0, u_{k-1}, \ldots, u_0, w_{k-1}, \ldots, w_0)$$

and the reformulated cost functional takes the form

$$E\{g_N(\tilde{x}_N)\}, \qquad \text{where} \qquad \tilde{x}_N = (x_0, \ldots, x_N, u_0, \ldots, u_{N-1}, w_0, \ldots, w_{N-1}).$$

The control law sought consists of functions μ_k of the present and past states $x_k, \ldots, x_0$, the past controls $u_{k-1}, \ldots, u_0$, and the past disturbances $w_{k-1}, \ldots, w_0$. Naturally, we must assume that past disturbances are known to the controller for otherwise we are faced with a problem with imperfect state information. The DP algorithm takes the form

$$\begin{aligned} J_{N-1}(x_0, \ldots, x_{N-1}, u_0, \ldots, u_{N-2}, w_0, \ldots, w_{N-2}) \\ = \min_{u_{N-1} \in U_{N-1}(x_{N-1})} E_{w_{N-1}} \{g_N(x_0, \ldots, x_{N-1}, f_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}), \\ u_0, \ldots, u_{N-1}, w_0, \ldots, w_{N-1})\}. \end{aligned}$$

$$\begin{aligned} J_k(x_0, \ldots, x_k, u_0, \ldots, u_{k-1}, w_0, \ldots, w_{k-1}) \\ = \min_{u_k \in U_k(x_k)} E_{w_k}\{J_{k+1}(x_0, \ldots, x_k, f_k(x_k, u_k, w_k), \\ u_0, \ldots, u_k, w_0, \ldots, w_k)\}, \qquad k = 0, \ldots, N-2. \end{aligned}$$

Similar algorithms may be written for the case where the control constraint set depends on past states or controls, and so on.

Correlated Disturbances

We turn now to the case where the disturbances w_k are correlated. Here we shall assume that the w_k are elements of a Euclidean space and that the probability distribution of w_k does not depend explicitly on the current state x_k and control u_k, but rather it depends explicitly on the prior values of the disturbances $w_0, \ldots, w_{k-1}$. By using statistical methods (see, e.g., [A1]) it is often possible to represent the process $w_0, w_1, \ldots, w_{N-1}$ by means of a linear system

$$\begin{aligned} y_{k+1} &= A_k y_k + \xi_k, \qquad k = 0, 1, \ldots, N-1, \quad y_0 = 0, \\ w_k &= C_k y_{k+1}, \end{aligned}$$

where A_k, C_k are matrices of appropriate dimension and ξ_k are independent random vectors with given distribution. In other words, the correlated process $w_0, \ldots, w_{N-1}$ is represented as the output of a linear system perturbed by a white process, that is, a process consisting of independent random vectors as shown in Figure 1.17. By considering now y_k as additional state variables, we have a new system equation:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k[x_k, u_k, C_k(A_k y_k + \xi_k)] \\ A_k y_k + \xi_k \end{bmatrix}. \tag{1.21}$$

By taking as the new state the pair $\bar{x}_k = (x_k, y_k)$ and as new disturbance the vector ξ_k, we can write (1.21) as

$$\bar{x}_{k+1} = \bar{f}_k(\bar{x}_k, u_k, \xi_k).$$

By suitable reformulation of the cost functional, the problem is reduced to the form of the basic problem. Note that it is necessary that y_k, $k = 1, \ldots, N-1$, can be observed by the controller in order for the problem to be one of perfect state information. This is true when the matrix C_{k-1} is the identity matrix and w_{k-1} is observable. The DP algorithm takes the form

$$J_N(x_N, y_N) = g_N(x_N),$$

$$J_k(x_k, y_k) = \min_{u_k \in U_k(x_k)} E_{\xi_k}\{g_k[x_k, u_k, C_k(A_k y_k + \xi_k)] + J_{k+1}[f_k[x_k, u_k, C_k(A_k y_k + \xi_k)], A_k y_k + \xi_k]\}.$$

When C_k is the identity matrix, the optimal controller is of the form

$$\{\mu_0^*(x_0), \mu_1^*(x_1, w_0), \ldots, \mu_{N-1}^*(x_{N-1}, w_{N-2})\}.$$

Forecasts

Finally, consider the case where at time k the decision maker has access to a forecast y_k that results in a reassessment of the probability distribution of w_k and possibly of future disturbances. For example, y_k may be an exact prediction of w_k or an exact prediction that the probability distribution of w_k is a specific one out of a finite collection of distributions. Forecasts that can be of interest in practice are, for example, probabilistic predictions on the state of the weather, the interest rate for money, and demand for inventory.

Generally, forecasts can be handled by state augmentation although

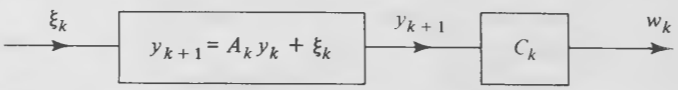


Figure 1.17 Representation of a correlated process $\{w_k\}$ as the output of a linear system driven by a white noise sequence $\{\xi_k\}$.

the reformulation into the form of the basic problem may be quite complex. We will treat here only a simple situation.

Consider the case where the probability distribution of w_k does not depend on x_k, u_k, w_{k-1}, . . ., w_0. Assume that at the beginning of each period k the decision maker receives an accurate prediction that the next disturbance w_k will be selected in accordance with a particular probability distribution out of a finite collection of given distributions $\{P_{k|1}, \ldots, P_{k|n}\}$; that is, if the forecast is i, then w_k is selected according to $P_{k|i}$. The a priori probability that the forecast at time k will be i is p_i^k and is given. Thus the forecasting process can be represented by means of the equation

$$y_{k+1} = \xi_k, \tag{1.22}$$

where y_{k+1} can take the values 1, 2, . . ., n and ξ_k is a random variable taking the values 1, 2, . . ., n with probabilities $p_1^{k+1}, \ldots, p_n^{k+1}$. The interpretation here is that when ξ_k takes the value i, then w_{k+1} will occur in accordance with the probability distribution $P_{k+1|i}$.

By combining the system equation and (1.22), we obtain an augmented system given by

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} f_k(x_k, u_k, w_k) \\ \xi_k \end{bmatrix} = \tilde{f}_k(\tilde{x}_k, u_k, \tilde{w}_k).$$

The new state is $\tilde{x}_k = (x_k, y_k)$ and the new disturbance is $\tilde{w}_k = (w_k, \xi_k)$. The probability distribution of $\tilde{w}_k$ is given in terms of the distributions $P_{k|i}$ and the probabilities p_i^k, and depends explicitly on $\tilde{x}_k$ (via y_k) but not on the prior disturbances $\tilde{w}_{k-1}, \ldots, \tilde{w}_0$. Thus by suitable reformulation of the cost functional, the problem can be cast into the framework of the basic problem. It is to be noted that the *control applied at each time is a function of both the current state and the current forecast.* The DP algorithm takes the form

$$J_N(x_N, y_N) = g_N(x_N),$$

$$J_k(x_k, y_k) = \min_{u_k \in U_k(x_k)} E_{w_k}\left\{ g_k(x_k, u_k, w_k) + \sum_{i=1}^{n} p_i^{k+1} J_{k+1}[f_k(x_k, u_k, w_k), i] \middle| y_k \right\},$$

$$k = 0, 1, \ldots, N-1,$$

where the expectation over w_k is taken with respect to the probability distribution P_{k,y_k}, where y_k may take the values 1, 2, . . ., n. Extension to forecasts covering several periods can be handled similarly, albeit at the expense of increased complexity. Problems where forecasts can be affected by the control action also admit a similar treatment.

It should be clear from the preceding discussion that state augmentation is a very general and potent device for reformulating problems of decision under uncertainty into the basic problem form. One should also realize that there are many ways to reformulate a problem by augmenting the state in different ways. The basic guideline is to *select as the augmented state*

at time k only those variables the knowledge of which can be of benefit to the decision maker when making the kth decision. For example, in the case of single period time lags it is intuitively obvious that the controller can benefit from knowing at time k the values of x_k, x_{k-1}, u_{k-1}, since these variables affect the value of the next state x_{k+1} through the system equation. The controller, however, has nothing to gain from knowing at time k the values of x_{k-2}, x_{k-3}, . . ., u_{k-2}, . . ., and for this reason these past states and controls need not be included in the augmented state, although their inclusion is technically possible. The theme of considering as state variables in the reformulated problem only those variables the knowledge of which would be beneficial to the decision making process will be predominant in the discussion of problems with imperfect state information (Chapter 3).

Finally, we note that whereas state augmentation is a convenient device, it tends to introduce both analytical and computational complexities, which in many cases are insurmountable.

1.6 NOTES

Dynamic programming is a simple mathematical technique that has been used for many years by engineers, mathematicians, and social scientists in a variety of contexts. It was Bellman, however, who realized in the early 1950s that DP could be developed (in conjunction with the then appearing digital computer) into a systematic tool for optimization. Bellman demonstrated the broad scope of DP and helped streamline its theory. His early books [B5, B6] are still popular reading. Other books related to DP are [H8], [H16], [K5], [K14], [N2], [R7], [W7], and [W11]. For a rigorous treatment of DP in general spaces that resolves the associated measurability issues and supplements the present text, see [B23]. For continuous-time formulations, see [B7] and [F3].

The connection of the Viterbi algorithm with the shortest path problem has been clarified in [O2] and [F4]. For further material on search methods and their use in game programs, see [P9]. For background on shortest paths, branch-and-bound, and combinatorial optimization see [P2].

As discussed in Section 1.1, the basic problem was formulated rigorously only for the case where the disturbance spaces are countable sets. *Nonetheless, the DP algorithm can often be utilized in a simple way when the countability assumption is not satisfied and there are further restrictions (such as measurability) on the class of admissible control laws.* The advanced reader will understand how this can be done by solving Problem 12, which shows that if one can find within a subset of control laws (such as those satisfying certain measurability restrictions) a control law that attains the minimum in the DP algorithm, then this control law is optimal. This fact may be used to establish rigorously many of our subsequent results concerning specific applications in Chapters 2 and 3. For example, in linear–quadratic

problems (Section 2.1) one determines from the DP equations a control law that is a linear function of the current state. When w_k can take uncountably many values, it is necessary that admissible control laws consist only of functions μ_k which are Borel measurable. Since the linear control law belongs to this class, the result of Problem 12 guarantees that this control law is optimal.

PROBLEMS

1. Use the DP algorithm to solve the following two problems:
   (a) minimize $\Sigma_{i=0}^{3} x_i^2 + u_i^2$
   subject to $x_0 = 0$, $x_4 = 8$, $u_i =$ nonnegative integer,
   $x_{i+1} = x_i + u_i$, $i = 0, 1, 2, 3$;
   (b) minimize $\Sigma_{i=0}^{3} x_i^2 + 2u_i^2$
   subject to $x_0 = 5$, $u_i \in \{0, 1, 2\}$,
   $x_{i+1} = x_i - u_i$, $i = 0, 1, 2, 3$.
2. Air transportation is available between n cities, in some cases directly and in others through intermediate stops and change of carrier. The air fare between cities i and j is denoted C_{ij} ($C_{ij} = C_{ji}$), and for notational convenience we write $C_{ij} = \infty$ if there is no direct flight between i and j. The problem is to find the cheapest possible air fare for going from any city i to any other city j perhaps through intermediate stops. Formulate a DP algorithm for solving this problem. Solve the problem for $n = 6$ and $C_{12} = 30$, $C_{13} = 60$, $C_{14} = 25$, $C_{15} = C_{16} = \infty$, $C_{23} = C_{24} = C_{25} = \infty$, $C_{26} = 50$, $C_{34} = 35$, $C_{35} = C_{36} = \infty$, $C_{45} = 15$, $C_{46} = \infty$, $C_{56} = 15$.
3. Suppose we have a machine that is either running or broken down. If it runs throughout one week, it makes a gross profit of \$100. If it fails during the week, gross profit is zero. If it is running at the start of the week and we perform preventive maintenance, the probability that it will fail during the week is 0.4. If we do not perform such maintenance, the probability of failure is 0.7. However, maintenance will cost \$20. When the machine is broken down at the start of the week, it may either be repaired at a cost of \$40, in which case it will fail during the week with a probability of 0.4, or it may be replaced at a cost of \$150 by a new machine that is guaranteed to run through its first week of operation. Find the optimal repair, replacement, and maintenance policy that maximizes total profit over four weeks, assuming a new machine at the start of the first week.
4. A game of the blackjack variety is played by two players as follows: Both players throw a die. The first player, knowing his opponent's result, may stop or may throw the die again and add the result to the result of his previous throw. He then may stop or throw again and add the result of the new throw to the sum of his previous throws. He may repeat this process as many times as he wishes. If his sum exceeds seven (i.e., he busts), he loses the game. If he stops before exceeding seven, the second player takes over and throws the die successively until the sum of his throws is four or higher. If the sum of the second player is over seven, he loses the game. Otherwise the player with

the larger sum wins, and in case of a tie the second player wins. The problem is to determine a stopping strategy for the first player that maximizes his probability of winning for each possible initial throw of the second player. Formulate the problem in terms of DP and find an optimal stopping strategy for the case where the second player's initial throw is three. *Hint:* Take $N = 6$ and a state space consisting of the following 15 states:

x^1: busted,

x^{1+i}: already stopped at sum i $(1 \leq i \leq 7)$,

x^{8+i}: current sum is i but the player has not yet stopped $(1 \leq i \leq 7)$.

The optimal strategy is to throw until the sum is four or higher.

5. *Min–Max Problems.* In the framework of the basic problem, consider the case where the disturbances $w_0, w_1, \ldots, w_{N-1}$ do not have a probabilistic description but rather are known to belong to corresponding given sets $W_k(x_k, u_k) \subset D_k$, $k = 0, 1, \ldots, N-1$, which may depend on the current state x_k and control u_k. Consider the problem of finding a control law $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ with $\mu_k(x_k) \in U_k(x_k)$ for all x_k, k, which minimizes the cost functional

$$J_\pi(x_0) = \max_{\substack{w_k \in W_k[x_k, \mu_k(x_k)] \\ k=0,1,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k[x_k, \mu_k(x_k), w_k] \right\}.$$

The DP algorithm for this problem takes the form

$$J_N(x_N) = g_N(x_N),$$

$$J_k(x_k) = \min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \{g_k(x_k, u_k, w_k) + J_{k+1}[f_k(x_k, u_k, w_k)]\}.$$

Assuming that $J_k(x_k) > -\infty$ for all x_k and k, show that the optimal cost equals $J_0(x_0)$. *Hint:* Imitate the proof for the stochastic case; prove and use the following fact: If U, W, X are three sets, $f : W \to X$ is a function, and M is the set of all functions $\mu : X \to U$, then for any functions $G_0 : W \to (-\infty, \infty]$, $G_1 : X \times U \to (-\infty, \infty]$ such that

$$\min_{u \in U} G_1[f(w), u] > -\infty, \qquad \text{for all } w \in W$$

we have

$$\min_{\mu \in M} \max_{w \in W} \{G_0(w) + G_1[f(w), \mu(f(w))]\} = \max_{w \in W} \{G_0(w) + \min_{u \in U} G_1[f(w), u]\}.$$

6. *Discounted Cost per Stage.* In the framework of the basic problem, consider the case where the cost functional is of the form

$$E\left\{ \alpha^N g_N(x_N) + \sum_{k=0}^{N-1} \alpha^k g_k(x_k, u_k, w_k) \right\},$$

where α is a discount factor with $0 < \alpha < 1$. Show that an alternate form of the DP algorithm is given by

$$V_N(x_N) = g_N(x_N),$$

$$V_k(x_k) = \min_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + \alpha V_{k+1}[f_k(x_k, u_k, w_k)]\}.$$

7. *Exponential Cost Functional.* In the framework of the basic problem, consider

the case where the cost functional is of the form

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ \exp\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right] \right\}.$$

(a) Show that the optimal cost and an optimal policy can be obtained from the last step of the DP algorithm

$$J_N(x_N) = \exp[g_N(x_N)],$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \underset{w_k}{E}\{J_{k+1}[f_k(x_k, u_k, w_k)] \exp[g_k(x_k, u_k, w_k)]\}.$$

Show that the algorithm yields an optimal control law if one exists.

(b) Define the functions $V_k(x_k) = \ln J_k(x_k)$. Assume also that g_k is a function of x_k and u_k only (and not of w_k). Show that the above DP algorithm can be rewritten

$$V_N(x_N) = g_N(x_N),$$

$$V_k(x_k) = \min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + \ln \underset{w_k}{E}\{\exp V_{k+1}[f_k(x_k, u_k, w_k)]\} \right].$$

8. *Terminating Process.* Consider the case in the basic problem where the system evolution terminates at time i when a given value $\overline{w}_i$ of the disturbance at time i occurs, or when a termination decision u_i is made by the controller. If termination occurs at time i, the resulting cost is

$$T + \sum_{k=0}^{i} g_k(x_k, u_k, w_k),$$

where T is a termination cost. If the process has not terminated up to the final time N, the resulting cost is $g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)$. Reformulate the problem into the framework of the basic problem. *Hint:* Augment the state space with a special termination state.

9. *Multiplicative Cost.* In the framework of the basic problem, consider the case where the cost functional has the multiplicative form

$$\underset{\substack{w_k \\ k=0,\ldots,N-1}}{E} \{g_N(x_N) \cdot g_{N-1}(x_{N-1}, u_{N-1}, w_{N-1}) \cdots g_0(x_0, u_0, w_0)\}.$$

Devise an algorithm of the DP type for this problem under the assumption $g_k(x_k, u_k, w_k) \geq 0$ for all x_k, u_k, w_k, and k.

10. Assume that we have a vessel whose maximum weight capacity is z and whose cargo is to consist of different quantities of N different items. Let v_i denote the value of the ith type of item, w_i the weight of ith type of item, and x_i the number of items of type i that are loaded in the vessel. The problem of determining the most valuable cargo is that of maximizing $\sum_{i=1}^{N} x_i v_i$ subject to the constraints $\sum_{i=1}^{N} x_i w_i \leq z$ and $x_i = 0, 1, 2, \ldots$. Formulate this problem in terms of DP.

11. Consider a device consisting of N stages connected in series, where each stage consists of a particular component. The components are subject to failure, and to increase the reliability of the device duplicate components are provided. For $j = 1, 2, \ldots, N$, let $(1 + m_j)$ be the number of components for the jth stage, let $p_j(m_j)$ be the probability of successful operation of the jth stage when

$(1 + m_j)$ components are used, and let c_j denote the cost of a single component at the jth stage. Consider the problem of finding the number of components at each stage that maximize the reliability of the device expressed by

$$p_1(m_1) \cdot p_2(m_2) \cdots p_N(m_N)$$

subject to the cost constraint $\sum_{j=1}^{N} c_j m_j \leq A$, where $A > 0$ is given. Formulate the problem in terms of DP.

12. *Minimization over a Subset of Policies.* This problem is primarily of theoretical interest (see the end of the Notes to this chapter). Consider a variation of the basic problem whereby we seek

$$\min_{\pi \in \tilde{\Pi}} J_\pi(x_0),$$

where $\tilde{\Pi}$ is some given *subset* of the set of sequences $\{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ of functions $\mu_k : S_k \to C_k$ with $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Assume that

$$\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$$

belongs to $\tilde{\Pi}$ and attains the minimum in the DP algorithm; that is, for all $k = 0, 1, \ldots, N-1$ and $x_k \in S_k$

$$J_k(x_k) = \mathop{E}_{w_k}\{g_k[x_k, \mu_k^*(x_k), w_k] + J_{k+1}[f_k(x_k, \mu_k^*(x_k), w_k)]\}$$

$$= \min_{u_k \in U_k(x_k)} \mathop{E}_{w_k}\{g_k(x_k, u_k, w_k) + J_{k+1}[f_k(x_k, u_k, w_k)]\},$$

with $J_N(x_N) = g_N(x_N)$. Assume further that the functions J_k are real valued and the preceding expectations are well defined and finite. Show that π^* is optimal within $\tilde{\Pi}$ and

$$J_0(x_0) = \min_{\pi \in \tilde{\Pi}} J_\pi(x_0) = J_{\pi^*}(x_0).$$

13. *Semilinear Systems.* Consider a problem involving the system

$$x_{k+1} = A_k x_k + f_k(u_k) + w_k,$$

where $x_k \in R^n$, f_k are given functions, and A_k and w_k are random $n \times n$ matrices and n-vectors, respectively, with given probability distributions that do not depend on x_k, u_k or prior values of A_k and w_k. Assume that the cost functional is of the form

$$\mathop{E}_{\substack{A_k, w_k \\ k=0,1,\ldots,N-1}} \left\{ c_N' x_N + \sum_{k=0}^{N-1} [c_k' x_k + g_k[\mu_k(x_k)]] \right\},$$

where c_k are given vectors and g_k given functions. Show that if the optimal cost for this problem is finite and the control constraint sets $U_k(x_k)$ are independent of x_k, then the cost-to-go functions of the DP algorithm are affine (linear plus constant). Assuming that there is at least one optimal policy, show that there exists an optimal policy that consists of constant functions μ_k^*; that is, $\mu_k^*(x_k) =$ constant for all $x_k \in R^n$.

14. A farmer annually producing x_k units of a certain crop stores $(1 - u_k)x_k$ units of his production, where $0 \leq u_k \leq 1$, and invests the remaining $u_k x_k$ units, thus increasing the next year's production to a level x_{k+1} given by

$$x_{k+1} = \lambda_k + w_k u_k x_k, \qquad k = 0, 1, \ldots, N-1.$$

The scalars w_k are independent random variables with identical probability

distributions that do not depend either on x_k or u_k. Furthermore, $E\{w_k\} = \overline{w} > 0$. The problem is to find the optimal investment policy that maximizes the total expected product stored over N years

$$\underset{\substack{w_k \\ k=0,1,\ldots,N-1}}{E} \left\{ x_N + \sum_{k=0}^{N-1} (1 - u_k)x_k \right\}.$$

Show that one optimal control law is given by:

(a) If $\overline{w} > 1$, $\mu_0^*(x_0) = \cdots = \mu_{N-1}^*(x_{N-1}) = 1$.

(b) If $0 < \overline{w} < 1/N$, $\mu_0^*(x_0) = \cdots = \mu_{N-1}^*(x_{N-1}) = 0$.

(c) If $1/N \leq \overline{w} \leq 1$,

$$\mu_0^*(x_0) = \cdots = \mu_{N-\bar{k}-1}^*(x_{N-\bar{k}-1}) = 1,$$

$$\mu_{N-\bar{k}}^*(x_{N-\bar{k}}) = \cdots = \mu_{N-1}^*(x_{N-1}) = 0,$$

where $\bar{k}$ is such that $1/(\bar{k} + 1) < \overline{w} \leq 1/\bar{k}$. (Note that this control law consists of constant functions.)

15. Let x_k denote the number of educators in a certain country at time k and let y_k denote the number of research scientists at time k. New scientists (potential educators or research scientists) are produced during the kth period by educators at a rate γ_k per educator, while educators and research scientists leave the field due to death, retirement, and transfer at a rate δ_k. The scalars γ_k, $k = 0, 1, \ldots, N - 1$, are independent identically distributed random variables taking values within a closed and bounded interval of positive numbers. Similarly δ_k, $k = 0, 1, \ldots, N - 1$, are independent identically distributed and take values in an interval $[\delta, \delta']$ with $0 < \delta \leq \delta' < 1$. By means of incentives a science policy maker can determine the proportion u_k of new scientists produced at time k who become educators. Thus the number of research scientists and educators evolves according to the equations

$$x_{k+1} = (1 - \delta_k)x_k + u_k\gamma_k x_k,$$
$$y_{k+1} = (1 - \delta_k)y_k + (1 - u_k)\gamma_k x_k.$$

The initial numbers x_0, y_0 are known, and it is required to find a policy $\{\mu_0^*(x_0, y_0), \ldots, \mu_{N-1}^*(x_{N-1}, y_{N-1})\}$ with

$$0 < \alpha \leq \mu_k^*(x_k, y_k) \leq \beta < 1, \qquad \text{for all } x_k, y_k, \text{ and } k,$$

which maximizes $E_{\gamma_k, \delta_k}\{y_N\}$ (i.e., the expected final number of research scientists after N periods). The scalars α and β are given.

(a) Show that the cost-to-go functions $J_k(x_k, y_k)$ are linear; that is, for some scalars λ_k, μ_k

$$J_k(x_k, y_k) = \lambda_k x_k + \mu_k y_k.$$

(b) Derive an optimal policy $\{\mu_0^*, \ldots, \mu_{N-1}^*\}$ under the assumption $E\{\gamma_k\} > E\{\delta_k\}$, and show that this optimal policy can consist of constant functions.

(c) Assume that the proportion of new scientists who become educators at time k is $u_k + \varepsilon_k$ (rather than u_k), where ε_k are identically distributed independent random variables that are also independent of γ_k, δ_k and take values in the interval $[-\alpha, 1 - \beta]$. Derive the form of the cost-to-go functions and the optimal policy.

16. *DP on Two Parallel Processors* [*L1*]. Formulate a DP algorithm to solve the

deterministic problem of Section 1.3 on a parallel computer with two processors. One processor should execute a forward algorithm and the other a backward algorithm.

17. The paragraphing problem deals with breaking up a sequence of N words $w_1, \ldots, w_N$ with lengths $L_1, \ldots, L_N$ into lines of length A. In a simple version of the problem, words are separated by blanks whose ideal width is b, but blanks can stretch or shrink if necessary, so that a line $w_i, w_{i+1}, \ldots, w_{i+k}$ has length exactly A. The cost associated with the line is $(k + 1)\,|b' - b|$, where $b' = (A - L_i - \cdots - L_{i+k})/(k + 1)$ is the actual average width of the blanks, except if we have the last line ($N = i + k$), in which case the cost is zero when $b' \geq b$. Formulate a DP algorithm for solving for the minimum cost separation. *Hint:* Consider the subproblems of optimally separating $w_i, \ldots, w_N$ for $i = 1, \ldots, N$.

18. *Computer Assignment.* In the classical game of blackjack the player draws cards knowing only one card of the dealer. The player loses upon reaching a sum of cards exceeding 21. If the player stops before exceeding 21, the dealer draws cards until reaching 17 or higher. The dealer loses upon reaching a sum exceeding 21 or a lower sum than the player's. If player and dealer end up with an equal sum no one wins, and in all other cases the dealer wins. An ace for the player may be counted as a 1 or an 11 as the player chooses. An ace for the dealer is counted as an 11 if this results in a sum from 17 to 21 and as a 1 otherwise. Jacks, queens, and kings count as 10 for both dealer and player. We assume an infinite card deck so the probability of a particular card showing up is independent of earlier cards.

(a) For every possible initial dealer card, calculate the probability that the dealer will reach a sum of 17, 18, 19, 20, 21, or over 21.

(b) Calculate the optimal choice of the player (draw or stop) for each of the possible combinations of dealer's card and player's sum of 12 to 20. Assume that the player's cards do not include an ace.

(c) Repeat part (b) for the case where the player's cards include an ace.

19. Consider a smaller version of a popular puzzle game. Three square tiles numbered 1, 2, and 3 are placed in a 2×2 grid with one space left empty. The two tiles adjacent to the empty space can be moved into that space, thereby creating new configurations. Use a DP argument to answer the question whether it is possible to generate a given configuration starting from any other configuration.

20. From a pile of eleven matchsticks, two players take turns removing one or four sticks. The player who removes the last stick wins. Use a DP argument to show that there is a winning strategy for the player who plays first.

21. *The Counterfeit Coin Problem.* We are given six coins, one of which is counterfeit and is known to be heavier or lighter than the rest. Construct a strategy to find the counterfeit coin using a two-pan scale in a minimum average number of tries. *Hint:* There are two initial decisions that make sense: (1) test two of the coins against two others, and (2) test one of the coins against one other.

22. Given a sequence of matrix multiplications

$$M_1 M_2 \cdots M_k M_{k+1} \cdots M_N,$$

where M_k, $k = 1, \ldots, N$, is of dimension $n_k \times n_{k+1}$, the order in which

multiplications are carried out can make a difference. For example, if $n_1 = 1$, $n_2 = 10$, $n_3 = 1$, and $n_4 = 10$, the calculation $((M_1M_2)M_3)$ requires 20 multiplications, but the calculation $(M_1(M_2M_3))$ requires 200 multiplications. Derive a DP algorithm for finding the optimal multiplication order. Solve the problem for $n = 3$, $n_1 = 2$, $n_2 = 10$, $n_3 = 5$, and $n_4 = 1$.

23. *Doubling Algorithms.* Consider a deterministic finite state problem that is time invariant in the sense that the state and control spaces, the cost per stage, and the system equation are the same for each time period. Let $J_k(x, y)$ be the optimal cost to reach state y at time k from state x at time 0. Show that for all k

$$J_{2k}(x, y) = \min_z \{J_k(x, z) + J_k(z, y)\}.$$

Discuss how this equation may be used with advantage to solve problems with a large number of stages.

24. *Complexity of DP for Shortest Paths.* Consider the shortest path algorithm

$$J_k(i) = \min_{j=1,\ldots,N} \{c_{ij} + J_{k+1}(j)\}$$

of Section 1.3. Suppose m is the largest number of arcs in a shortest path from any node $1, \ldots, N$ to the destination node t. Show that the algorithm can be terminated after m steps and that the number of arithmetic operations required is bounded by γmL, where L is the number of arcs and γ is a number that is independent of m, L, and N.

25. *Monotonicity Property of DP.* An evident, yet very important property of the DP algorithm is that if the terminal cost g_N is changed to a uniformly larger cost $\bar{g}_N$ [i.e., $g_N(x_N) \leq \bar{g}_N(x_N)$ for all x_N], then the corresponding costs $J_k(x_k)$ will be uniformly increased. More generally, given two functions J_{k+1} and $\bar{J}_{k+1}$ with $J_{k+1}(x_{k+1}) \leq \bar{J}_{k+1}(x_{k+1})$ for all x_{k+1}, we have, for all x_k and $u_k \in U_k(x_k)$,

$$\begin{aligned} &\underset{w_k}{E}\{g_k(x_k, u_k, w_k) + J_{k+1}[f_k(x_k, u_k, w_k)]\} \\ &\leq \underset{w_k}{E}\{g_k(x_k, u_k, w_k) + \bar{J}_{k+1}[f_k(x_k, u_k, w_k)]\}. \end{aligned}$$

Suppose now that in the basic problem the system and cost are time invariant; that is, $S_k \equiv S$, $C_k \equiv C$, $D_k \equiv D$, $f_k \equiv f$, $U_k(x_k) \equiv U(x_k)$, and $g_k \equiv g$. Show that if in the DP algorithm we have $J_{N-1}(x) \leq J_N(x)$ for all $x \in S$ then

$$J_k(x) \leq J_{k+1}(x), \qquad \text{for all } x \in S \text{ and } k.$$

Similarly, if we have $J_{N-1}(x) \geq J_N(x)$ for all $x \in S$, then

$$J_k(x) \geq J_{k+1}(x), \qquad \text{for all } x \in S \text{ and } k.$$

26. Modify the forward search algorithm of Section 1.4 so that it simultaneously finds the shortest paths from the origin s to several destination nodes and also detects when shortest paths do not exist. *Hint:* Connect the destination nodes with a new artificial node using arcs with very large length.

27. *Dijkstra's Algorithm for Shortest Paths.* Consider the best-first version of the forward search algorithm of Section 1.4. Here at each iteration we select a node j from OPEN that has minimum estimate d_j over all nodes in OPEN.

(a) Show that each node j will enter OPEN at most once and show that at

the time it enters CLOSED its estimate d_j is equal to the shortest distance from s to j.

(b) Show that the number of arithmetic operations required for termination is bounded by cN^2 where N is the number of nodes and c is some constant.

28. *Distributed Asynchronous Shortest Path Computation* [B19]. Consider the problem of finding a shortest path from nodes 1, 2, . . ., N to node t, and assume that all arc lengths c_{ij} are positive. Consider the iteration

$$\begin{aligned} d_i^{k+1} &= \min_j \{c_{ij} + d_j^k\}, \qquad i = 1, 2, \ldots, N, \\ d_t^{k+1} &= 0. \end{aligned} \tag{1.23}$$

(a) It was shown in Section 1.3 that, if the initial condition is $d_i^0 = \infty$ for $i = 1, \ldots, N$ and $d_t^0 = 0$, then (1.23) yields the shortest distances d_i^* in N steps. Show that if the initial condition is $d_i^0 = 0$, for all $i = 1, \ldots, N$, t, then (1.23) yields the shortest distances in a finite number of steps. Provide an upper bound for this number in terms of the problem data.

(b) Assume that the iteration

$$d_i := \min_j \{c_{ij} + d_j\} \tag{1.24}$$

is executed at node i in parallel with the corresponding iteration for d_j at every other node j. However, the times of execution of this iteration at the various nodes are not synchronized. Furthermore, each node i communicates the results of its latest computation of d_i at arbitrary times with potentially large communication delays. Therefore, there is the possibility of a node executing iteration (1.24) several times before receiving a communication from every other neighboring node. Assume that each node never stops executing iteration (1.24) and transmitting the result to the other nodes. Show that the estimates d_i^T available at time T at the corresponding nodes i equal the shortest distances d_i^* for all T after a finite time $\overline{T}$. *Hint:* Let $\overline{d}_i^k$ and $\underline{d}_i^k$ be the estimates generated by (1.23) when starting from the first and the second initial conditions in part (a), respectively. Show that for every k there exists a time T_k such that for all $T \geq T_k$ we have $\underline{d}_i^k \leq d_i^T \leq \overline{d}_i^k$. For a detailed analysis of asynchronous iterative algorithms, including algorithms for shortest paths and dynamic programming, see D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods" Prentice-Hall, 1989.

References

[A1] Anderson, B. D. O., and Moore, J. B., *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J., 1979.

[A2] Aoki, M., *Optimization of Stochastic Systems—Topics in Discrete-Time Systems*. Academic Press, New York, 1967.

[A3] Aoki, M., *Introduction to Optimization Techniques*. Macmillan, New York, 1971.

[A4] Aoki, M., and Li, M. T., Optimal discrete-time control systems with cost for observation, *IEEE Trans. Automatic Control* **AC-14** (1969), 165–175.

[A5] Arrow, K. J., *Aspects of the Theory of Risk Bearing*. Yrjo Jahnsson Lecture Series, Helsinki, Finland, 1965.

[A6] Arrow, K. J., Blackwell, D., and Girshick, M. A., Bayes and minimax solutions of sequential design problems, *Econometrica* **17** (1949), 213–244.

[A7] Arrow, K. J., Harris, T., and Marschack, J., Optimal inventory policy, *Econometrica* **19** (1951), 250–272.

[A8] Arrow, K. J., Karlin, S., and Scarf, H., *Studies in the Mathematical Theory of Inventory and Production*. Stanford University Press, Stanford, California, 1958.

[A9] Ash, R. B., *Basic Probability Theory*. Wiley, New York, 1970.

[A10] Ash, R. B., and Gardner, M. F., *Topics in Stochastic Processes*. Academic Press, New York, 1975.

[A11] Aström, K. J., *Introduction to Stochastic Control Theory*. Academic Press, New York, 1970.

[A12] Aström, K. J., Theory and applications of adaptive control—a survey, *Automatica,* **19** (1983), 471–486.

[A13] Aström, K. J., and Wittenmark, B. *Computer Controlled Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1984.

[A14] Aström, K. J., and Wittenmark, B., On self-tuning regulators, *Automatica* **9** (1973), 185–199.

[A15] Atkinson, R. C., Bower, G. H., and Crothers, E. J., *An Introduction to Mathematical Learning Theory*. Wiley, New York, 1965.

[B1] Bar-Shalom, Y., and Tse, E., Dual effect, certainty equivalence, and separation in stochastic control, *IEEE Trans. Automatic Control* **AC-19** (1974), 494–500.

[B2] Baras, J. S., Dorsey, A. J., and Makowski, A. M., Two competing queues with linear costs: The μc-rule is often optimal, Report SRR 83-1, Department of Electrical Engineering, University of Maryland, 1983.

[B3] Baras, J. S., and Dorsey, A. J., Stochastic control of two partially observed competing queues, *IEEE Trans. Aut. Control* **AC-26** (1981), 1106–1117.

[B4] Bather, J., Optimal decision procedures for finite Markov chains, *Advances in Appl. Probability* **5** (1973), 328–339, 521–540, 541–553.

[B5] Bellman, R., *Dynamic Programming*. Princeton University Press, Princeton, N.J., 1957.

[B6] Bellman, R., and Dreyfus, S., *Applied Dynamic Programming*. Princeton University Press, Princeton, N.J., 1962.

[B7] Bensoussan, A., and Lions, J. L., *Applications des Inequations Variationnelles en Controle Stochastique*. Dunod, Paris, 1978.

[B8] Bertsekas, D. P., On the separation theorem for linear systems, quadratic criteria, and correlated noise, unpubl. rep., Electronic Systems Lab., MIT, Cambridge, Mass., September 1970.

[B9] Bertsekas, D. P., Control of uncertain systems with a set-membership description of the uncertainty." Ph.D. Dissertation, MIT, Cambridge, Mass., 1971.

[B10] Bertsekas, D. P., Stochastic optimization problems with nondifferentiable cost functionals with an application in stochastic programming, *Proc. IEEE Decision and Control Conf.*, New Orleans, La., December 1972.

[B11] Bertsekas, D. P., On the solution of some minimax problems, *Proc. IEEE Decision and Control Conf.*, New Orleans, La., December 1972.

[B12] Bertsekas, D. P., Infinite time reachability of state space regions by using feedback control, *IEEE Trans. Automatic Control* **AC-17** (1972), 604–613.

[B13] Bertsekas, D. P., Stochastic optimization problems with nondifferentiable cost functionals, *J. Optimization Theory Appl.* **12** (1973), 218–231.

[B14] Bertsekas, D. P., Linear convex stochastic control problems over an infinite time horizon, *IEEE Trans. Automatic Control* **AC-18** (1973), 314–315.

[B15] Bertsekas, D. P., Necessary and sufficient conditions for existence of an optimal portfolio, *J. Econom. Theory* **8** (1974), 235–247.

[B16] Bertsekas, D. P., Monotone mappings with application in dynamic programming. *SIAM J. Control and Optimization* **15** (1977), 438–464.

[B17] Bertsekas, D. P., On error bounds for successive approximation methods, *IEEE Trans. Automatic Control* **AC-21** (1976), 394–396.

[B18] Bertsekas, D. P., Convergence of discretization procedures in dynamic programming, *IEEE Trans. Automatic Control* **AC-20** (1975), 415–419.

[B19] Bertsekas, D. P., Distributed dynamic programming, *IEEE Trans. Automatic Control* **AC-27** (1982), 610–616.

[B20] Bertsekas, D. P., and Castanon, D., Dynamic Aggregation Methods for Discounted Markovian Decision Problems, Proc. of 25th IEEE Conference on Decision and Control, Athens, Greece, Dec. 1986; *IEEE Trans. on Aut. Control* (to appear).

[B21] Bertsekas, D. P., and Rhodes, I. B., On the minimax reachability of target sets and target tubes, *Automatica* **7** (1971), 233–247.

[B22] Bertsekas, D. P., and Rhodes, I. B., Sufficiently informative functions and the minimax feedback control of uncertain dynamic systems, *IEEE Trans. Automatic Control* **AC-18** (1973), 117–124.

[B23] Bertsekas, D. P., and Shreve, S. E., *Stochastic Optimal Control: The Discrete Time Case*. Academic Press, New York, 1978.

[B24] Bertsekas, D. P., and Shreve, S. E., Existence of optimal stationary policies in deterministic optimal control, *J. Math. Anal. and Appl.* **69** (1979), 607–620.

[B25] Billingsley, P., The singular function of bold play, *American Scientist* **71** (1983), 392–397.

[B26] Blackwell, D., Discrete dynamic programming, *Ann. Math. Statist.* **33** (1962), 719–726.

[B27] Blackwell, D., Discounted dynamic programming, *Ann. Math. Statist.* **36** (1965), 226–235.

[B28] Blackwell, D., Positive dynamic programming, *Proc. 5th Berkeley Symp. Math., Statist., and Probability* **1** (1965), 415–418.

[B29] Blackwell, D., On stationary policies, *J. Roy. Statist. Soc. Ser. A*, **133** (1970), 33–38.

[B30] Blackwell, D., and Girshick, M. A., *Theory of Games and Statistical Decisions*. Wiley, New York, 1954.

[B31] Blake, I. F., and Thomas, J. B., On a class of processes arising in linear estimation theory, *IEEE Trans. Information Theory* **IT-14** (1968), 12–16.

[B32] Borkar, V., and Varaiya, P. P., Identification and adaptive control of Markov chains, *SIAM J. on Control and Optimization* **20** (1982), 470–489.

[B33] Borkar, V., and Varaiya, P. P., Adaptive control of Markov chains, I: Finite parameter set, *IEEE Trans. Aut. Control* **AC-24** (1979), 953–958.

[B34] Brown, B. W., On the iterative method of dynamic programming on a finite space discrete Markov process, *Ann. Math. Statist.* **36** (1965), 1279–1286.

[C1] Chernoff, H., *Sequential Analysis and Optimal Design*. Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, Pa., 1972.

[C2] Chung, K. L., *Markov Chains with Stationary Transition Probabilities*. Springer-Verlag, Berlin and New York, 1960.

[C3] Cooper, C. A., and Nahi, N. E., An optimal stochastic control problem with

observation cost, *Proc. Joint Automatic Control Conf.*, Atlanta, Ga., June 1970.

[C4] Courcoubetis, C., and Varaiya, P. P., The service process with least thinking time maximizes resource utilization, *IEEE Trans. Aut. Control* **AC-29** (1984), 1005–1008.

[C5] Crabill, T. B., Gross, D., and Magazine, M. J., A classified bibliography of research on optimal design and control of queues, *Operations Res.* **25** (1977), 219–232.

[D1] DeGroot, M. H., *Optimal Statistical Decisions.* McGraw-Hill, New York, 1970.

[D2] Denardo, E. V., Contraction mappings in the theory underlying dynamic programming, *SIAM Rev.* **9** (1967), 165–177.

[D3] D'Epenoux, F., Sur un probleme de production et de stockage dans l'aleatoire, *Rev. Francaise Automat. Informat. Recherche Operationnelle* **14** (1960) (English Transl.: *Management Sci.* **10** (1963), 98–108).

[D4] Derman, C., *Finite State Markovian Decision Processes.* Academic Press, New York, 1970.

[D5] Deshpande, J. G., Upadhyay, T. N., and Lainiotis, D. G., Adaptive control of linear control systems, *Automatica* **9** (1973), 107–115.

[D6] Dial, R., and others, A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees, *Networks* **9** (1979), 215–248.

[D7] Doshi, B., and Shreve, S., Strong consistency of a modified maximum likelihood estimator for controlled Markov chains, *J. of Applied Probability* **17** (1980), 726–734.

[D8] Dreyfus, S. E., *Dynamic Programming and the Calculus of Variations.* Academic Press, New York, 1965.

[D9] Dubins, L., and Savage, L. M., *How to Gamble If You Must.* McGraw-Hill, New York, 1965.

[D10] Dynkin, E. B., Controlled random sequences, *Theor. Probability Appl.* **10** (1965), 1–14.

[D11] Dynkin, E. B., and Juskevic, A. A., *Controlled Markov Processes.* Springer-Verlag, New York, 1979.

[E1] Eaton, J. H., and Zadeh, L. A., Optimal pursuit strategies in discrete state probabilistic systems, *Trans. ASME Ser. D. J. Basic Eng.* **84** (1962), 23–29.

[E2] Eckles, J. E., Optimum maintenance with incomplete information, *Operations Res.* **16** (1968), 1058–1067.

[E3] Ephremides, A., Varaiya, P. P., and Walrand, J. C., A simple dynamic routing problem, *IEEE Trans. Aut. Control* **AC-25** (1980), 690–693.

[F1] Federgruen, A., Schweitzer, P. J., and Tijms, H. C., Denumerable undiscounted semi-Markov decision processes with unbounded rewards, *Math. of Operations Res.* **8** (1983), 298–313.

[F2] Feller, W., *An Introduction to Probability Theory and Its Applications*. 3rd ed. Wiley, New York, 1968.

[F3] Fleming, W., and Rishel, R., *Optimal Deterministic and Stochastic Control*. Springer-Verlag, New York, 1975.

[F4] Forney, G. D., The Viterbi algorithm, *Proc. IEEE* **61** (1973), 268–278.

[F5] Fox, B. L., Finite state approximations to denumerable state dynamic programs, *J. Math. Anal. Appl.* **34** (1971), 665–670.

[G1] Gittins, J. C., Bandit processes and dynamic allocation indices, *J. Roy. Statist. Soc., B*, **41** (1979), 148–164.

[G2] Gittins, J. C., and Jones, D. M., A dynamic allocation index for the sequential design of experiments, in *Progress in Statistics* (J. Gani, ed.). North-Holland, Amsterdam, 1974, pp. 241–266.

[G3] Goodwin, G. C., and Sin, K. S. S., *Adaptive Filtering, Prediction, and Control*. Prentice-Hall, Englewood Cliffs, N.J., 1984.

[G4] Groen, G. J., and Atkinson, R. C., Models for optimizing the learning process, *Psychol. Bull.* **66** (1966), 309–320.

[G5] Gunckel, T. L., and Franklin, G. R., A general solution for linear sampled-data control, *Trans. ASME Ser. D. J. Basic Engrg.* **85** (1963), 197–201.

[H1] Hajek, B., Optimal control of two interacting service stations, *IEEE Trans. Aut. Control* **29** (1984), 491–499.

[H2] Hajek, B., and van Loon, T., Decentralized dynamic control of a multiaccess broadcast channel, *IEEE Trans. Aut. Control* **AC-27** (1982), 559–569.

[H3] Harrison, J. M., A priority queue with discounted linear costs, *Operations Res.* **23** (1975), 260–269.

[H4] Harrison, J. M., Dynamic scheduling of a multiclass queue: Discount optimality, *Operations Res.* **23** (1975), 270–282.

[H5] Hastings, N. A. J., Some notes on dynamic programming and replacement, *Operational Res. Quart.* **19** (1968), 453–464.

[H6] Haurie, A., and L'Ecuyer, P., Approximation and bounds in discrete event dynamic programming, *IEEE Trans. Aut. Control* **AC-31** (1986), 227–235.

[H7] Hendrikx, M., Van Nunen, J., and Wessels, J., On iterative optimization of structured Markov decision-processes, *Math. Oper. u. Statist.* **15** (1984), 439–459.

[H8] Heyman, D. P., and Sobel, M. J., *Stochastic Models in Operations Research*. McGraw-Hill, New York, Vol. I, 1982, Vol. II, 1984.

[H9] Hinderer, K., *Foundations of Non-Stationary Dynamic Programming with Discrete Time Parameter*. Springer-Verlag, New York, 1970.

[H10] Hoffman, K., and Kunze, R., *Linear Algebra*. Prentice-Hall, Englewood Cliffs, N.J., 1961.

[H11] Holt, C. C., Modigliani, F., and Simon, H. A., A linear decision rule for production and employment scheduling, *Management Sci.* **2** (1955), 1–30.

[H12] Hordijk, A., and Tijms, H., The method of successive approximations and Markovian decision problems, *Operations Res.* **22** (1974), 519–521.

[H13] Hordijk, A., and Tijms, H. C., Convergence results and approximations for optimal (*s*, *S*) policies, *Management Sci.* **20** (1974), 1432–1438.

[H14] Hordijk, A., and Tijms, H. C., On a conjecture of Iglehart, *Management Sci.* **21** (1975), 1342–1345.

[H15] Howard, R., *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Mass., 1960.

[H16] Howard, R., *Dynamic Probabilistic Systems*, Vols. I and II. Wiley, New York, 1971.

[I1] *IEEE Trans. Aut. Control*, special issue on Linear–Quadratic Gaussian Problem, **AC-16** (1971).

[I2] Iglehart, D. L., Optimality of (*S*, *s*) policies in the infinite horizon dynamic inventory problem, *Management Sci.* **9** (1963), 259–267.

[I3] Iglehart, D. L., Dynamic programming and stationary analysis of inventory problems, *in* Scarf, H., Gilford, D., and Shelly, M. (eds.), *Multistage Inventory Models and Techniques*. Stanford University Press, Stanford, Calif., 1963.

[J1] Jacobson, D. H., Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games, *IEEE Trans. Aut. Control* **AC-18** (1973), 124–131.

[J2] Jacobson, D. H., A general result in stochastic optimal control of nonlinear discrete-time systems with quadratic performance criteria, *J. Math. Anal. Appl.* **47** (1974), 153–161.

[J3] Jazwinski, A. H., *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.

[J4] Jewell, W., Markov renewal programming I and II, *Operations Res.* **11** (1963), 938–971.

[J5] Joseph, P. D., and Tou, J. T., On linear control theory, *AIEE Trans.* **80** (II) (1961), 193–196.

[K1] Kalman, R. E., A new approach to linear filtering and prediction problems, *Trans. ASME Ser. D. J. Basic Engrg.* **82** (1960), 35–45.

[K2] Kalman, R. E., and Koepcke, R. W., Optimal synthesis of linear sampling control systems using generalized performance indexes, *Trans. ASME* **80** (1958), 1820–1826.

[K3] Karush, W., and Dear, E. E., Optimal stimulus presentation strategy for a stimulus sampling model of learning, *J. Mathematical Psychology* **3** (1966), 15–47.

[K4] Katehakis, M., and Veinott, A. F., *The Multi-Armed Bandit Problem: Decomposition and Computation*, T.R. 41, Department of Operations Research, Stanford University, Stanford, Calif., July 1985.

[K5] Kaufmann, A., and Cruon, R., *Dynamic Programming*. Academic Press, New York, 1967.

[K6] Kemeny, J. G., and Snell, J. L., *Finite Markov Chains*. Van Nostrand-Reinhold, New York, 1960.

[K7] Kimemia, J., Gershwin, S. B., and Bertsekas, D. P., Computation of production control policies by a dynamic programming technique, in *Analysis and Optimization of Systems* (A. Bensoussan and J. L. Lions, eds.), Springer-Verlag, New York, 1982, pp. 243–269.

[K8] Kimemia, J., Hierarchical control of production in flexible manufacturing systems, Ph.D. thesis, MIT, Department of Electrical Engineering and Computer Science, April 1982.

[K9] Kleinman, D. L., On an iterative technique for Riccati equation computations, *IEEE Trans. Aut. Control,* **AC-13** (1968), 114–115.

[K10] Kumar, P. R., A survey of some results in stochastic adaptive control, *SIAM J. Control Optimization* **23** (1985), 329–380.

[K11] Kumar, P. R., Optimal adaptive control of linear–quadratic–Gaussian systems, *SIAM J. Control Optimization* **21** (1983), 163–178.

[K12] Kumar, P. R., and Lin, W., Optimal adaptive controllers for unknown Markov chains, *IEEE Trans. Aut. Control* **AC-27** (1982), 765–774.

[K13] Kumar, P. R., and Varaiya, P. P., *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice-Hall, Englewood Cliffs, N.J., 1986.

[K14] Kushner, H. J., *Introduction to Stochastic Control*. Holt, Rinehart and Winston, New York, 1971.

[K15] Kushner, H. J., Optimality conditions for the average cost per unit time problem with a diffusion model, *SIAM J. Control Optimization* **16** (1978), 330–346.

[L1] Lasserre, J. B., A mixed forward–backward dynamic programming method using parallel computation, *J. Optimization Theory Appl.* **45** (1985), 165–168.

[L2] Levy, D., *The Chess Computer Handbook*. B. T. Batsford Ltd., London, 1984.

[L3] Lin, W., and Kumar, P. R., Optimal control of a queueing system with two heterogeneous servers, *IEEE Trans. Aut. Control* **AC-29** (1984), 696–703.

[L4] Lippman, S., Applying a new device in the optimization of exponential queuing systems, *Operations Res.* **23** (1975), 687–710.

[L5] Liusternik, L., and Sobolev, V., *Elements of Functional Analysis*. Ungar, New York, 1961.

[L6] Ljung, L., On positive real transfer functions and the convergence of some recursions, *IEEE Trans. Aut. Control* **AC-22** (1977), 539–551.

[L7] Ljung, L., *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, N.J., 1986.

[L8] Ljung, L., and Soderstrom, T., *Theory and Practice of Recursive Identification*. MIT Press, Cambridge, Mass., 1983.

[L9] Luenberger, D. G., *Optimization by Vector Space Methods*. Wiley, New York, 1969.

[L10] Luenberger, D. G., *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Mass., 1984.

[M1] Malinvaud, E., First order certainty equivalence, *Econometrica* **37** (1969), 706–718.

[M2] Mandl, P., Estimation and control in Markov chains, *Advances in Applied Probability* **6** (1974), 40–60.

[M3] Manne, A., Linear programming and sequential decisions, *Management Sci.* **6** (1960), 259–267.

[M4] Markovitz, H., *Portfolio Selection*. Wiley, New York, 1959.

[M5] McQueen, J., A modified dynamic programming method for Markovian decision problems, *J. Math. Anal. Appl.* **14** (1966), 38–43.

[M6] Meditch, J. S. *Stochastic Optimal Linear Estimation and Control*. McGraw-Hill, New York, 1969.

[M7] Mendel, J. M., *Discrete Techniques of Parameter Estimation—The Equation Error Formulation*. Dekker, New York, 1973.

[M8] Morton, T. E., On the asymptotic convergence rate of cost differences for Markovian decision processes, *Operations Res.* **19** (1971), 244–248.

[M9] Morton, T. E., and Wecker, W., Discounting, ergodicity and convergence for Markov decision processes, *Management Sci.* **23** (1977), 890–900.

[M10] Mossin, J., Optimal multi-period portfolio policies, *J. Business* **41** (1968), 215–229.

[N1] Nahi, N., *Estimation Theory and Applications*. Wiley, New York, 1969.

[N2] Nemhauser, G. L., *Introduction to Dynamic Programming*. Wiley, New York, 1966.

[N3] Newborn, M., *Computer Chess*. Academic Press, New York, 1975.

[N4] Norman, J. M., Dynamic programming in tennis—When to use a fast serve, *J. Opl. Res. Soc.* **36** (1985), 75–77.

[O1] Odoni, A. R., On finding the maximal gain for Markov decision processes, *Operations Res.* **17** (1969), 857–860.

[O2] Omura, J. K., On the Viterbi decoding algorithm, *IEEE Trans. Information Theory* **IT-15** (1969), 177–179.

[O3] Ornstein, D., On the existence of stationary optimal strategies, *Proc. Amer. Math. Soc.* **20** (1969), 563–569.

[O4] Ortega, J. M., and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York, 1970.

[P1] Pallu de la Barriere, R., *Optimal Control Theory*. Saunders, Philadelphia, 1967.

[P2] Papadimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, N.J., 1982.

[P3] Papadimitriou, C. H., and Tsitsiklis, J. N., The complexity of Markov decision processes, *Math. of Operations Research* (to appear).

[P4] Pape, V., Implementation and efficiency of Moore algorithms for the shortest path problem, *Math. Progr.* **7** (1974), 212–222.

[P5] Papoulis, A., *Signal Analysis*. McGraw-Hill, New York, 1977.

[P6] Papoulis, A., *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, 1965.

[P7] Parzen, E., *Modern Probability Theory and Its Applications*. Wiley, New York, 1960.

[P8] Pattipati, K. R., and Kleinman, D. L., Priority assignment using dynamic programming for a class of queueing systems, *IEEE Trans. Aut. Control* **AC-26** (1981), 1095–1106.

[P9] Pearl, J., *Heuristics*. Addison-Wesley, Reading, Mass., 1984.

[P10] Platzman, L., Comments on "Optimal and suboptimal stationary controls for Markov chains," *IEEE Trans. Aut. Control* **AC-24** (1979), 375.

[P11] Platzman, L., Improved conditions for convergence in undiscounted Markov renewal programming, *Operations Res.* **25** (1977), 529–533.

[P12] Popyak, J. L., Brown, R. L., and White, C. C., Discrete versions of an algorithm due to Varaiya, *IEEE Trans. Aut. Control* **AC-24** (1979), 503–504.

[P13] Porteus, E., Bounds and transformations for finite Markov decision chains, *Operations Res.* **23** (1975), 761–784.

[P14] Porteus, E., Some bounds for discounted sequential decision processes, *Management Sci.* **18** (1971), 7–11.

[P15] Porteus, E., Overview of iterative methods for discounted finite Markov and semi-Markov decision chains, in *Rec. Developments in Markov Decision Processes*, R. Hartley, L. C. Thomas, and D. J. White (eds.), Academic Press, London, 1980.

[P16] Puterman, M. L. (ed.), *Dynamic Programming and Its Applications*, Academic Press, New York, 1978.

[P17] Puterman, M. L., and Brumelle, S. L., The analytic theory of policy iteration, in *Dynamic Programming and Its Applications*, M. L. Puterman (ed.), Academic Press, New York, 1978.

[P18] Puterman, M. L., and Shin, M. C., Action elimination procedures for modified policy iteration algorithms, *Operations Res.* **30** (1982), 301–318.

[P19] Puterman, M. L., and Shin, M. C., Modified policy iteration algorithms for discounted Markov decision problems, *Management Sci.* **24** (1978), 1127–1137.

[R1] Rivest, R. L., Network control by Bayesian broadcast, MIT, LCS Report, August 1985.

[R2] Rockafellar, R. T., *Convex Analysis*. Princeton University Press, Princeton, N.J., 1970.

[R3] Rockafellar, R. T., and Wets, R., Stochastic convex programming: Basic duality, *Pacific J. Math.* **62** (1976), 173–195.

[R4] Rosberg, Z., Varaiya, P. P., and Walrand, J. C., Optimal control of service in tandem queues, *IEEE Trans. Aut. Control* **AC–27** (1982), 600–609.

[R5] Ross, S. M., Arbitrary state Markovian decision processes, *Ann. Math. Statist.* **39** (1968), 2118–2122.

[R6] Ross, S. M., *Applied Probability Models with Optimization Applications.* Holden-Day, San Francisco, 1970.

[R7] Ross, S. M., *Introduction to Stochastic Dynamic Programming.* Academic Press, New York, 1983.

[R8] Ross, S. M., *Stochastic Processes.* Wiley, New York, 1983.

[R9] Royden, H. L., *Real Analysis.* Macmillan, New York, 1968.

[R10] Rudin, E., *Principles of Mathematical Analysis.* McGraw-Hill, New York, 1964.

[S1] Saridis, G. N., *Self-Organizing Control of Stochastic Systems.* Dekker, New York, 1977.

[S2] Saridis, G. N., and Dao, T. K., A learning approach to the parameter-adaptive self-organizing control problem, *Automatica* **8** (1972), 589–597.

[S3] Sawaragi, Y., and Yoshikawa, T., Discrete-time Markovian decision processes with incomplete state observation, *Ann. Math. Statist.* **41** (1970), 78–86.

[S4] Scarf, H., The optimality of (s, S) policies for the dynamic inventory problem, *Proceedings of the 1st Stanford Symposium on Mathematical Methods in the Social Sciences.* Stanford University Press, Stanford, Calif., 1960.

[S5] Schal, M., On the optimality of (s, S) policies in dynamic inventory models with finite horizon, *SIAM J. Appl. Math.* **30** (1976), 528–537.

[S6] Schal, M., Conditions for optimality in dynamic programming and for the limit of n-stage optimal policies to be optimal, *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete* **32** (1975), 179–196.

[S7] Schweitzer, P. J., Perturbation theory and finite Markov chains, *J. Appl. Prob.* **5** (1968), 401–413.

[S8] Schweitzer, P. J., Bottleneck determination in networks of queues, Graduate School of Management Working Paper No. 8107, University of Rochester, Rochester, N.Y., February 1981.

[S9] Schweitzer, P. J., Data transformations for Markov renewal programming, talk at National ORSA Meeting, Atlantic City, N.Y., November 1972.

[S10] Schweitzer, P. J., Iterative solution of the functional equations of undiscounted Markov renewal programming, *J. Math. Anal. Appl.* **34** (1971), 495–501.

[S11] Schweitzer, P. J., Puterman, M. L., and Kindle, K. W., Iterative aggregation–disaggregation procedures for solving discounted semi-Markovian reward processes, *Operations Research* **33** (1985), 589–605.

[S12] Schweitzer, P. J., and Federgruen, A., The asymptotic behavior of value iteration in Markov decision problems, *Math. Operations Res.* **2** (1977), 360–381.

[S13] Schweitzer, P. J., and Federgruen, A., The functional equations of undiscounted Markov renewal programming, *Math. Operations Res.* **3** (1978), 308–321.

[S14] Serfozo, R., An equivalence between discrete and continuous time Markov decision processes, *Operations Res.* **27** (1979), 616–620.

[S15] Serfozo, R., Optimal control of random walks, birth and death processes, and queues, *Adv. Appl. Prob.* **13** (1981), 61–83.

[S16] Shannon, C., Programming a digital computer for playing chess, *Phil. Mag.* **41** (1950), 356–375.

[S17] Shapley, L. S., Stochastic games, *Proc. Nat. Acad. Sci. U.S.A.* **39** (1953).

[S18] Sharpe, W., *Portfolio Theory and Capital Markets*, McGraw-Hill, New York, 1970.

[S19] Simon, H. A., Dynamic programming under uncertainty with a quadratic criterion function, *Econometrica* **24** (1956), 74–81.

[S20] Smallwood, R. D., The analysis of economic teaching strategies for a simple learning model, *J. Math. Psychology* **8** (1971), 285–301.

[S21] Smallwood, R. D., and Sondik, E. J., The optimal control of partially observable Markov processes over a finite horizon, *Operations Res.* **11** (1973), 1071–1088.

[S22] Sobel, M. J., The optimality of full-service policies, *Operations Res.* **30** (1982), 636–649.

[S23] Sondik, E. J., The optimal control of partially observable Markov processes, Ph.D. Dissertation, Department of Engineering–Economic Systems, Stanford University, Stanford, Calif., June 1971.

[S24] Stein, G., and Saridis, G. N., A parameter adaptive control technique, *Automatica* **5** (1969), 731–739.

[S25] Stidham, S., and Prabhu, N. U., Optimal control of queuing systems, in *Mathematical Methods in Queuing Theory* (Lecture Notes in Economics and Math. Syst., Vol. 98), A. B. Clarke (Ed.), Springer-Verlag, New York, 1974, pp. 263–294.

[S26] Stidham, S. S., Optimal control of admission to a queueing system, *IEEE Trans. Aut. Control* **AC-30** (1985), 705–713.

[S27] Strang, G., *Linear Algebra and Its Applications*, Academic Press, New York, 1976.

[S28] Strauch, R., Negative dynamic programming, *Ann. Math. Statist.* **37** (1966), 871–890.

[S29] Striebel, C. T., Sufficient statistics in the optimal control of stochastic systems, *J. Math. Anal. Appl.* **12** (1965), 576–592.

[S30] Striebel, C. T., *Optimal Control of Discrete Time Stochastic Systems*, Springer-Verlag, New York, 1975.

[S31] Sussman, R., Optimal control of systems with stochastic disturbances, Electronics Research Lab., University of California, Berkeley, Rep. No. 63-20, November 1963.

[S32] Sworder, D. D., Bayes' controllers with memory for a linear system with jump parameters, *IEEE Trans. Auto. Control* **AC-17** (1972), 119–121.

[T1] Thau, F. E., and Witsenhausen, H. S., A comparison of closed-loop and open-loop optimum systems, *IEEE Trans. Auto. Control* **AC-11** (1966), 619–621.

[T2] Theil, H., Econometric models and welfare maximization, *Weltwirtsch. Arch.* **72** (1954), 60–83.

[T3] Tse, E., and Athans, M., Adaptive stochastic control for a class of linear systems, *IEEE Trans. Auto. Control* **AC-17** (1972), 38–52.

[T4] Tse, E., and Bar-Shalom, Y., An actively adaptive control for linear systems with random parameters via the dual control approach, *IEEE Trans. Auto. Control* **AC-18** (1973), 109–117.

[T5] Tse, E., Bar-Shalom, Y., and Meier, L., III, Wide-sense adaptive dual control for nonlinear stochastic systems, *IEEE Trans. Auto. Control* **AC-18** (1973), 98–108.

[T6] Tsitsiklis, J. N., Convexity and characterization of optimal policies in a dynamic routing problem, *J. Optimization Theory Appl.* **44** (1984), 105–136.

[T7] Tsitsiklis, J. N., Periodic review inventory systems with continuous demand and discrete order sizes, *Management Sci.* **30** (1984), 1250–1254.

[T8] Tsitsiklis, J. N., *Analysis of a Multiaccess Control Scheme*, Lab. for Info. and Decision Systems Report LIDS-P-1534, MIT, Feb. 1986.

[T9] Tsitsiklis, J. N., A lemma on the multiarmed bandit problem, *IEEE Trans. Aut. Control* **AC-31** (1986), 576–577.

[V1] Vajda, S., Stochastic programming, Chapter 14 *in* Abadie, J. (ed.), *Integer and Nonlinear Programming*, North-Holland, Amsterdam, 1970.

[V2] Varaiya, P. P., Optimal and suboptimal stationary controls of Markov chains, *IEEE Trans. Aut. Control* **AC-23** (1978), 388–394.

[V3] Varaiya, P. P., Walrand, J. C., and Buyukkoc, C., Extensions of the multiarmed bandit problem: The discounted case, *IEEE Trans. Auto. Control* **AC-30** (1985), 426–439.

[V4] Veinott, A. F., Jr., On finding optimal policies in discrete dynamic programming with no discounting, *Ann. Math. Statist.* **37** (1966), 1284–1294.

[V5] Veinott, A. F., Jr., The status of mathematical inventory theory, *Management Sci.* **12** (1966), 745–777.

[V6] Veinott, A. F., Jr., Discrete dynamic programming with sensitive discount optimality criteria, *Ann. Math. Statist.* **40** (1969), 1635–1660.

[V7] Veinnott, A. F., Jr., The optimal inventory policy for batch ordering, *Operations Res.* **13** (1965), 424–432.

[V8] Veinott, A. F., Jr., and Wagner, H. M., Computing optimal (s, S) policies, *Management Sci.* **11** (1965), 525–552.

[V9] Vershik, A. M., Some characteristic properties of Gaussian stochastic processes, *Theory Probability Appl.* **9** (1964), 353–356.

[W1] Wald, A., *Sequential Analysis*. Wiley, New York, 1947.

[W2] Walkup, D., and Wets, R., Stochastic programs with recourse, *SIAM J. Appl. Math.* **15** (1967), 1299–1314.

[W3] Weiss, G., and Pinedo, M., Scheduling tasks with exponential service times on nonidentical processors to minimize various cost functions, *J. Appl. Prob.* **17** (1980), 187–202.

[W4] White, C. C., and Harrington, D. P., Application of Jensen's inequality to adaptive suboptimal design, *J. Optimization Theory Appl.* **32** (1980), 89–99.

[W5] White, C. C., and Schlussel, K., Suboptimal design for large scale, multimodule systems, *Operations Res.* **29** (1981), 865–875.

[W6] White, D. J., Dynamic programming, Markov chains, and the method of successive approximations, *J. Math. Anal. Appl.* **6** (1963), 373–376.

[W7] White, D. J., *Dynamic Programming.* Holden-Day, San Francisco, 1969.

[W8] White, D. J., Finite state approximations for denumerable state infinite horizon discounted Markov decision processes: The method of successive approximations, in *Recent Developments in Markov Decision Processes*, Hartley, R., Thomas, L. C., and White, D. J. (eds.), Academic Press, New York, 1980, pp. 57–72.

[W9] Whitt, W., Approximations of dynamic programs I, *Math. Oper. Res.* **3** (1978), 231–243.

[W10] Whitt, W., Approximations of dynamic programs II, *Math. Oper. Res.* **4** (1979), 179–185.

[W11] Whittle, P., *Optimization over Time.* Wiley, New York, Vol. 1, 1982, Vol. 2, 1983.

[W12] Whittle, P., *Prediction and Regulation by Linear Least-Square Methods.* English Universities Press, London, 1963.

[W13] Witsenhausen, H. S., Minimax control of uncertain systems, Ph.D. Dissertation, Department of Electrical Engineering, MIT, Cambridge, Mass., May 1966.

[W14] Witsenhausen, H. S., Inequalities for the performance of suboptimal uncertain systems, *Automatica* **5** (1969), 507–512.

[W15] Witsenhausen, H. S., On performance bounds for uncertain systems, *SIAM J. Control* **8** (1970), 55–89.

[W16] Witsenhausen, H. S., Separation of estimation and control for discrete-time systems, *Proc. IEEE* **59** (1971), 1557–1566.

[Z1] Zangwill, W. I., *Nonlinear Programming: A Unified Approach.* Prentice-Hall, Englewood Cliffs, N.J., 1969.