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

1.1 INTRODUCTION

This book deals with situations where decisions are made in stages. The outcome of each decision is not fully predictable but can be anticipated to some extent before the next decision is made. The objective is to minimize a certain cost – a mathematical expression of what is considered an undesirable outcome.

A key aspect of such situations is that decisions cannot be viewed in isolation since one must balance the desire for low present cost with the undesirability of high future costs. The dynamic programming technique captures this tradeoff. At each stage, decisions are ranked based on the sum of the present cost and the expected future cost, assuming optimal decision making for subsequent stages.

There is a very broad variety of practical problems that can be treated by dynamic programming. In this book, we try 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 six chapters; its infinite horizon version will be the subject of the last chapter and Vol. II.

Our basic model has two principal features: (1) an underlying *discrete-time dynamic system*, and (2) a *cost function 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,

w_k is a random parameter (also called disturbance or noise depending on the context),

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

The cost function is additive in the sense that the cost incurred at time k, denoted by $g_k(x_k, u_k, w_k)$, accumulates over time. The total cost 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 an optimization of the *expected cost*

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

where the expectation is with respect to the joint distribution of the random variables involved. The optimization is over the controls $u_0, u_1, \ldots, u_{N-1}$, but some qualification is needed here; each control u_k is selected with some knowledge of the current state x_k (either its exact value or some other information relating to it).

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

Example 1.1 (Inventory Control)

Consider a problem of ordering a quantity of a certain item at each of N 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 $w_0, w_1, \ldots, w_{N-1}$ 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 equation

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

where negative stock corresponds to backlogged demand (see Fig. 1.1.1).

The cost incurred in period k consists of two components:

(a) A cost $r(x_k)$ representing a penalty for either positive stock x_k (holding cost for excess inventory) or negative stock x_k (shortage cost for unfilled demand).

(b) The purchasing cost cu_k, where c is cost per unit ordered.

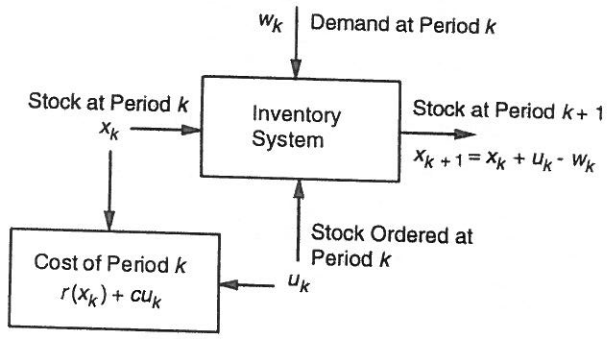

Figure 1.1.1 Inventory control example. At period k, the current stock (state) x_k, the stock ordered (control) u_k, and the demand (random disturbance) w_k determine the cost $r(x_k) + cu_k$ and the stock $x_{k+1} = x_k + u_k - w_k$ at the next period.

There is also a terminal cost $R(x_N)$ for being left with inventory x_N at the end of N periods. Thus, the total cost over N periods is

$$E\left\{R(x_N) + \sum_{k=0}^{N-1}\big(r(x_k) + cu_k\big)\right\}.$$

We want to minimize this cost by proper choice of the orders $u_0, \ldots, u_{N-1}$, subject to the natural constraint $u_k \geq 0$ for all k.

At this point we need to distinguish between *closed-loop* and *open-loop* minimization of the cost. In open-loop minimization we select all orders $u_0, \ldots, u_{N-1}$ at once at time 0, without waiting to see the subsequent demand levels. In closed-loop minimization we postpone placing the order u_k until the last possible moment (time k) when the current stock x_k will be known. The idea is that since there is no penalty for delaying the order u_k up to time k, we can take advantage of information that becomes available between times 0 and k (the demand and stock level in past periods).

Closed-loop optimization is of central importance in dynamic programming and is the type of optimization that we will consider almost exclusively in this book. Thus, in our basic formulation, decisions are made in stages while gathering information between stages that will be used to enhance the quality of the decisions. The effect of this on the structure of the resulting optimization problem is quite profound. In particular, in closed-loop inventory optimization we are not interested in finding optimal numerical values of the orders but rather we want to find an *optimal rule for selecting 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, in closed-loop inventory optimization, we want to find a sequence of functions μ_k, $k = 0, \ldots, N-1$, mapping stock x_k into order u_k so as to minimize the expected cost. The meaning of μ_k is that, for each k

and each possible value of x_k,

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

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

$$J_\pi(x_0) = E\left\{R(x_N) + \sum_{k=0}^{N-1}\big(r(x_k) + c\mu_k(x_k)\big)\right\},$$

and we want to minimize $J_\pi(x_0)$ for a given x_0 over all π that satisfy the constraints of the problem. This is a typical dynamic programming problem. We will analyze this problem in various forms in subsequent sections. For example, we will show in Section 4.2 that for a reasonable choice of the cost function, the optimal ordering policy is of the form

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

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:

(a) 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 the inventory example $f_k(x_k, u_k, w_k) = x_k + u_k - w_k$.

(b) *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 inventory example, we can think of a situation where the level of demand w_k is influenced by the current stock level x_k.

(c) A *control constraint*; in the example, we have $u_k \geq 0$. 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 $u_k \leq B - x_k$.

(d) 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 are some functions; in the example, we have $g_N(x_N) = R(x_N)$, and $g_k(x_k, u_k, w_k) = r(x_k) + cu_k$.

(e) *Optimization over (closed-loop) policies*, that is, rules for choosing u_k for each k and possible value of x_k.

Discrete-State and Finite-State Systems

In the preceding example, the state x_k was a continuous real variable, and it is easy to think of multidimensional generalizations where the state is an n-dimensional vector of real variables. It is also possible, however, that the state takes values from a discrete set, such as the integers.

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 $p_{ij}(u,k)$, which is the probability at time k that the next state will be j, given that the current state is i, and the control selected is u, i.e.,

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

Such a system can be described alternatively in terms of the discrete-time system equation

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

where the probability distribution of the random parameter w_k is

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

Conversely, given a discrete-state system in the form

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

together with the probability distribution $P_k(w_k \mid x_k, u_k)$ of w_k, we can provide an equivalent transition probability description. The corresponding transition probabilities are given by

$$p_{ij}(u,k) = P_k\{W_k(i,u,j) \mid x_k = i, u_k = u\},$$

where $W(i,u,j)$ is the set

$$W_k(i,u,j) = \{w \mid j = f_k(i,u,w)\}.$$

Thus a discrete-state system can equivalently be described in terms of a difference equation and in terms of transition probabilities. Depending on the given problem, it may be notationally more convenient to use one description over the other.

The following three examples illustrate discrete-state systems.

Example 1.2 (Machine Replacement)

Consider a problem of operating efficiently over N time periods a machine that can be in any one of n states, denoted $1, 2, \ldots, n$. The implication here is that state i is better than state $i+1$, and state 1 corresponds to a machine in perfect condition. In particular, we denote by $g(i)$ the operating cost per period when the machine is in state i, and we assume that

$$g(1) \leq g(2) \leq \cdots \leq g(n).$$

During a period of operation, the state of the machine can become worse or it may stay unchanged. We thus assume that the transition probabilities

$$p_{ij} = P\{\text{next state will be } j \mid \text{current state is } i\}$$

satisfy

$$p_{ij} = 0 \qquad \text{if } j < i.$$

We assume that at the start of each period we know the state of the machine and we must choose one of the following two options:

(a) Let the machine operate one more period in the state it currently is.

(b) Repair the machine and bring it to the perfect state 1 at a cost R.

We assume that the machine, once repaired, is guaranteed to stay in state 1 for one period. In subsequent periods, it may deteriorate to states $j > 1$ according to the transition probabilities p_{1j}.

Thus the objective here is to decide on the level of deterioration (state) at which it is worth paying the cost of machine repair, thereby obtaining the benefit of smaller future operating costs. Note that the decision should also be affected by the period we are in. For example, we would be less inclined to repair the machine when there are few periods left.

The system equation for this problem is represented by the graphs of Fig. 1.1.2. These graphs depict the transition probabilities between various pairs of states for each value of the control and are known as *transition probability graphs* or simply *transition graphs*. Note that there is a different graph for each control; in the present case there are two controls (repair or not repair).

Example 1.3 (Control of a Queue)

Consider a queueing system with room for n customers operating over N time periods. We assume that service of a customer can start (end) only at the beginning (end) of the period and that the system can serve only one customer at a time. The probability p_m of m customer arrivals during a 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

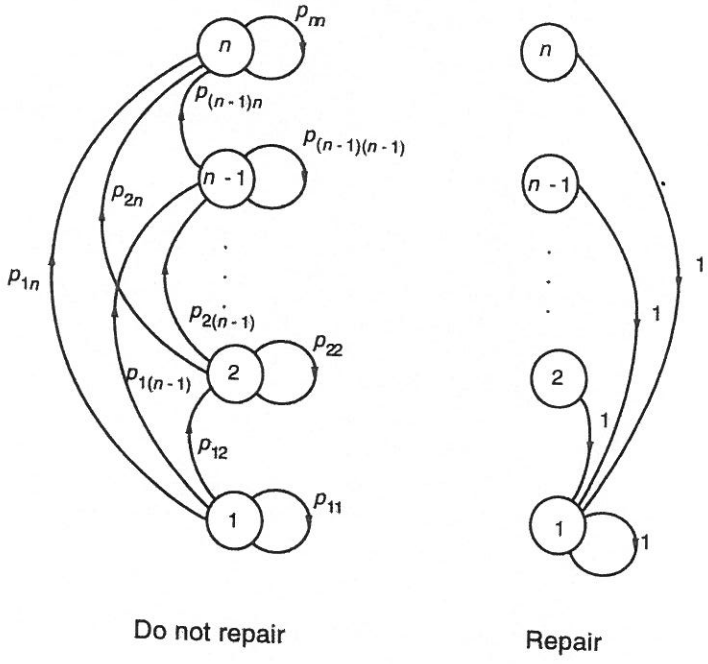

Figure 1.1.2 Machine replacement example. Transition probability graphs for each of the two possible controls (repair or not repair). At each stage and state i, the cost of repairing is $R+g(1)$, and the cost of not repairing is $g(i)$. The terminal cost is 0.

slow at the beginning of each period. With fast (slow) service, a customer in service at the beginning of a 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 $r(i)$ for each period for which there are i customers in the system. There is also a terminal cost $R(i)$ for i customers left in the system at the end of the last period. The problem is to choose, at each period, the type of service as a function of the number of customers in the system 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 control the type of service provided. The cost per period then is $r(i)$ 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 the period, the probability that the next state is j is independent of the type 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

$$p_{ij}(u_f) = 0, \qquad \text{if } j < i-1,$$

$$p_{ij}(u_f) = q_f p_0, \qquad \text{if } j = i-1,$$

$$\begin{aligned} p_{ij}(u_f) &= P\{j-i+1 \text{ arrivals, service completed}\} \\ &\quad + P\{j-i \text{ arrivals, service not completed}\} \\ &= q_f p_{j-i+1} + (1-q_f)p_{j-i}, \qquad \text{if } i-1 < j < n-1, \end{aligned}$$

$$p_{i(n-1)}(u_f) = q_f \sum_{m=n-i}^{\infty} p_m + (1-q_f)p_{n-1-i},$$

$$p_{in}(u_f) = (1-q_f) \sum_{m=n-i}^{\infty} p_m.$$

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.

Example 1.4 (Optimizing a Chess Match Strategy)

A player is about to play a two-game chess match with an opponent, and wants to maximize his winning chances. Each game can have one of two outcomes:

(a) A win by one of the players (1 point for the winner and 0 for the loser).

(b) A draw (1/2 point for each of the two players).

If the score is tied at 1-1 at the end of the two games, the match goes into sudden-death mode, whereby the players continue to play until the first time one of them wins a game (and the match). The player has two playing styles and he can choose one of the two at will in each game, independently of the style he chose in previous games.

(1) *Timid play* with which he draws with probability $p_d > 0$, and he loses with probability $(1-p_d)$.

(2) *Bold play* with which he wins with probability p_w, and he loses with probability $(1-p_w)$.

Thus, in a given game, timid play never wins, while bold play never draws. The player wants to find a style selection strategy that maximizes his probability of winning the match. Note that once the match gets into sudden death, the player should play bold, since with timid play he can at best prolong the

sudden death play, while running the risk of losing. Therefore, there are only two decisions for the player to make, the selection of the playing strategy in the first two games. Thus, we can model the problem as one with two stages, and with states the possible scores at the start of each of the first two stages (games), as shown in Fig. 1.1.3. The initial state is the initial score 0-0. The transition probabilities for each of the two different controls (playing styles) are also shown in Fig. 1.1.3. There is a cost at the terminal states: a cost of -1 at the winning scores 2-0 and 1.5-0.5, a cost of 0 at the losing scores 0-2 and 0.5-1.5, and a cost of $-p_w$ at the tied score 1-1 (since the probability of winning in sudden death is p_w). Note that to maximize the probability P of winning the match, we must minimize $-P$.

This problem has an interesting feature. One would think that if $p_w < 1/2$, the player would have a less than 50-50 chance of winning the match, even with optimal play, since his probability of losing is greater than his probability of winning any one game, regardless of his playing style. This is not so, however, because the player can adapt his playing style to the current score, but his opponent does not have that option. In other words, the player can use a closed-loop strategy, and it will be seen later that with optimal play, as determined by the dynamic programming algorithm, he has a better than 50-50 chance of winning the match provided p_w is higher than a threshold value $\overline{p}$, which, depending on the value of p_d, may satisfy $\overline{p} < 1/2$.

1.2 THE BASIC PROBLEM

We now formulate a general problem of decision under stochastic uncertainty over a finite number of stages. This problem, which we call *basic*, is the central problem in this book. We will discuss solution methods for this problem based on dynamic programming in the first six chapters, and we will extend our analysis to versions of this problem involving an infinite number of stages in the last chapter and in Vol. II of this work.

The basic problem is very general. In particular, we will not require that the state, control, or random parameter take a finite number of values or belong to 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, control, and random parameter spaces. For this reason it is convenient to proceed without any assumptions on the structure of these spaces; indeed such assumptions would become a serious impediment later.

Basic Problem

We are given a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1, \tag{2.1}$$

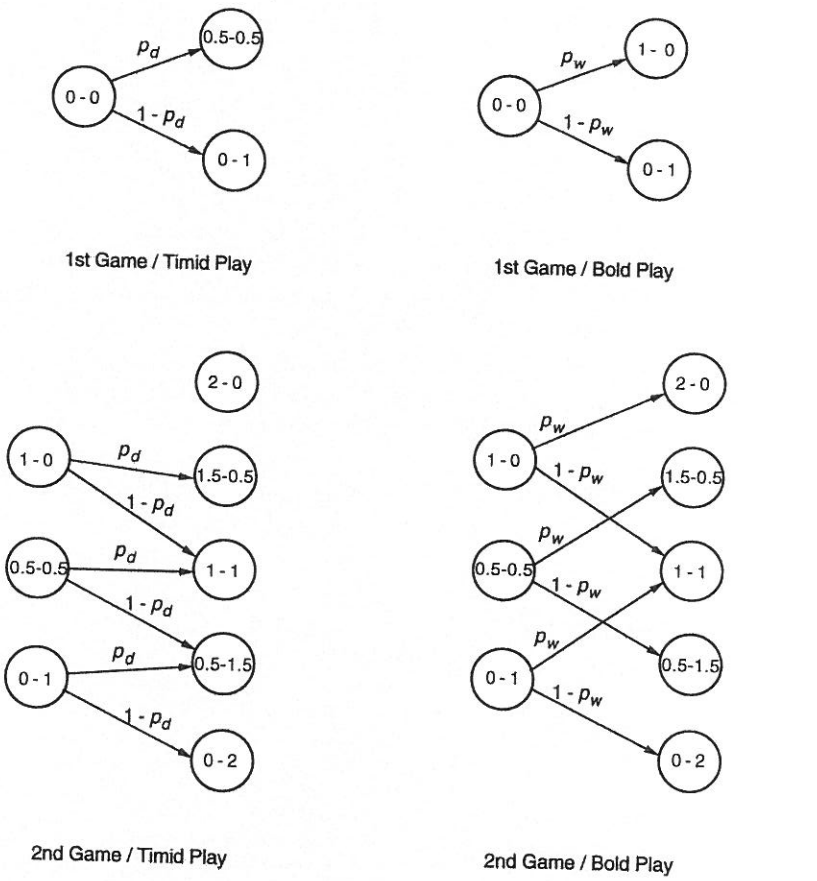

Figure 1.1.3 Chess match example. Transition probability graphs for each of the two possible controls (timid or bold play). Note here that the state space is not the same at each stage. The terminal cost is -1 at the winning final scores 2-0 and 1.5-0.5, 0 at the losing final scores 0-2 and 0.5-1.5, and $-p_w$ at the tied score 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 in a given nonempty subset $U(x_k) \subset C_k$, which depends on the current state x_k; that is, $u_k \in U_k(x_k)$ for all $x_k \in S_k$ and k.

The random disturbance w_k is characterized by a probability distribution $P_k(\cdot \mid x_k, u_k)$ 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 policies (also called control laws) that consist

of a sequence of functions

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

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 policies will be called *admissible*.

Given an initial state x_0 and an admissible policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, the system equation

$$x_{k+1} = f_k\big(x_k, \mu_k(x_k), w_k\big), \qquad k = 0, 1, \ldots, N-1, \tag{2.2}$$

makes x_k and w_k random variables with well-defined distributions. Thus, for given functions g_k, $k = 0, 1, \ldots, N$, the expected cost

$$J_\pi(x_0) = \mathop{E}_{\substack{w_k \\ k=0,1,\ldots,N-1}} \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right\} \tag{2.3}$$

is a well-defined quantity. For a given initial state x_0, an optimal policy π^* is one that minimizes this cost; that is,

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

where Π is the set of all admissible policies.

Note that the optimal policy π^* is associated with a fixed initial state x_0. However, an interesting aspect of the basic problem and of dynamic programming is that it is typically possible to find a policy π^* that is simultaneously optimal for all initial states.

The optimal cost depends on x_0 and is denoted by $J^*(x_0)$; that is,

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

It is useful to 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 it will not lead to any confusion.

The Role and Value of Information

We mentioned earlier the distinction between open-loop minimization, where we select all controls $u_0, \ldots, u_{N-1}$ at once at time 0, and closed-loop minimization, where we select a policy $\{\mu_0, \ldots, \mu_{N-1}\}$ that applies the control $\mu_k(x_k)$ at time k with knowledge of the current state x_k (see Fig. 1.2.1). With closed-loop policies, it is possible to achieve lower cost, essentially by taking advantage of the extra information (the value of the current state). The reduction in cost may be called the *value of the information* and can be significant indeed. If the 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 example of the preceding section, 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.

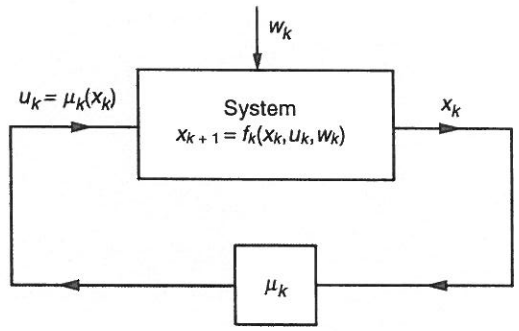

Figure 1.2.1 Information gathering in the basic problem. At each time k the controller observes the current state x_k and applies a control $u_k = \mu_k(x_k)$ that depends on that state.

Example 2.1

To illustrate the benefits of the proper use of information, let us consider the chess match example of the preceding section. There, a player can select timid play (probabilities p_d and $1 - p_d$ for a draw and a loss, respectively) or bold play (probabilities p_w and $1 - p_w$ for a win and a loss, respectively) in each of the two games of the match. Suppose the player chooses a policy of playing timid if and only if he is ahead in the score, as illustrated in Fig. 1.2.2; we will see in the next section that this policy is optimal, assuming $p_d > p_w$. Then after the first game (in which he plays bold), the score is 1-0 with probability p_w and 0-1 with probability $1 - p_w$. In the second game, he

plays timid in the former case and bold in the latter case. Thus after two games, the probability of a match win is $p_w p_d$, the probability of a match loss is $(1-p_w)^2$, and the probability of a tied score is $p_w(1-p_d)+(1-p_w)p_w$, in which case he has a probability p_w of winning the subsequent sudden-death game. Thus the probability of winning the match with the given strategy is

$$p_w p_d + p_w\big(p_w(1-p_d)+(1-p_w)p_w\big),$$

which, with some rearrangement, gives

$$\text{Probability of a match win} = p_w^2(2-p_w)+p_w(1-p_w)p_d. \tag{2.4}$$

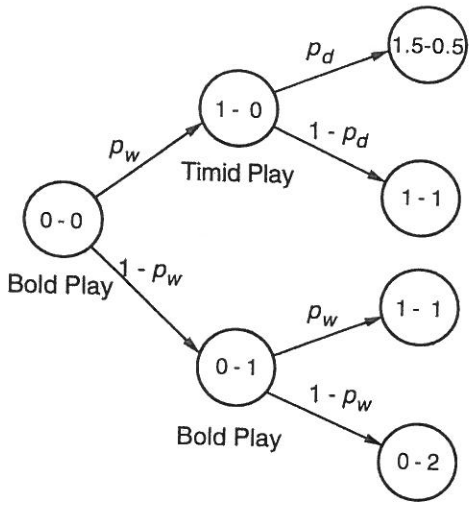

Figure 1.2.2 Illustration of the policy used in Example 2.1 to obtain a greater than 50-50 chance of winning the chess match and associated transition probabilities. The player chooses a policy of playing timid if and only if he is ahead in the score.

Suppose now that $p_w < 1/2$. Then the player has a greater probability of losing than winning any one game, regardless of the type of play he uses. From this we can infer that no open-loop strategy can give the player a greater than 50-50 chance of winning the match. Yet from Eq. (2.4) it can be seen that with the closed-loop strategy of playing timid if and only if the player is ahead in the score, the chance of a match win can be greater than 50-50, provided that p_w is close enough to $1/2$ and p_d is close enough to 1. As an example, for $p_w = 0.45$ and $p_d = 0.9$, Eq. (2.4) gives a match win probability of roughly 0.53.

To calculate the value of information, let us consider the four open-loop policies, whereby we decide on the type of play to be used without waiting to see the result of the first game. These are:

(1) Play timid in both games; this has a probability $p_d^2 p_w$ of winning the match.

(2) Play bold in both games; this has a probability $p_w^2 + p_w^2(1-p_w) = p_w^2(2-p_w)$ of winning the match.

(3) Play bold in the first game and timid in the second game; this has a probability $p_w p_d + p_w^2(1-p_d)$ of winning the match.

(4) Play timid in the first game and bold in the second game; this also has a probability $p_w p_d + p_w^2(1-p_d)$ of winning the match.

The first policy is always dominated by the others, and the optimal open-loop probability of winning the match is

$$\begin{aligned}\text{Open-loop probability of win} &= \max\big(p_w^2(2-p_w),\, p_w p_d + p_w^2(1-p_d)\big)\\ &= p_w^2 + p_w(1-p_w)\max(p_w, p_d).\end{aligned}$$

By making the reasonable assumption $p_d > p_w$, we see that the optimal open-loop policy is to play timid in one of the two games and play bold in the other, and that

$$\text{Open-loop probability of a match win} = p_w p_d + p_w^2(1-p_d). \tag{2.5}$$

For $p_w = 0.45$ and $p_d = 0.9$, Eq. (2.5) gives an optimal open-loop match win probability of roughly 0.425. Thus, the value of the information (the outcome of the first game) is the difference of the optimal closed-loop and open-loop values, which is approximately $0.53 - 0.425 = 0.105$. More generally, by subtracting Eqs. (2.4) and (2.5), we see that

$$\begin{aligned}\text{Value of information} &= p_w^2(2-p_w) + p_w(1-p_w)p_d - \big(p_w p_d + p_w^2(1-p_d)\big)\\ &= p_w^2(1-p_w).\end{aligned}$$

It should be noted, however, that whereas availability of the state information cannot hurt, it may not result in an advantage either. For instance, in deterministic problems, where no random disturbances are present, one can predict the future states given the initial state and the sequence of controls. Thus, 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 can be true even in some stochastic problems (see for example Exercise 1.13). 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 although the proof is technically complicated (see [BeS78]). The intuitive reason is that, for a given time k and state x_k, all future expected costs depend explicitly just on x_k and not on prior history.