# 17 MAKING COMPLEX DECISIONS

*In which we examine methods for deciding what to do today, given that we may decide again tomorrow.*

In this chapter, we address the computational issues involved in making decisions. Whereas Chapter 16 was concerned with one-shot or episodic decision problems, in which the utility of each action's outcome was well known, we will be concerned here with **sequential decision problems**, in which the agent's utility depends on a sequence of decisions. Sequential decision problems, which include utilities, uncertainty, and sensing, generalize the search and planning problems described in Parts II and IV. Section 17.1 explains how sequential decision problems are defined, and Sections 17.2 and 17.3 explain how they can be solved to produce optimal behavior that balances the risks and rewards of acting in an uncertain environment. Section 17.4 extends these ideas to the case of partially observable environments, and Section 17.5 develops a complete design for decision-theoretic agents in partially observable environments, combining dynamic Bayesian networks from Chapter 15 with decision networks from Chapter 16.

EQUENTIAL ECISION ROBLEMS

The second part of the chapter covers environments with multiple agents. In such environments, the notion of optimal behavior becomes much more complicated by the interactions among the agents. Section 17.6 introduces the main ideas of **game theory**, including the idea that rational agents might need to behave randomly. Section 17.7 looks at how multiagent systems can be designed so that multiple agents can achieve a common goal.

## 17.1 SEQUENTIAL DECISION PROBLEMS

### An example

Suppose that an agent is situated in the $4 \times 3$ environment shown in Figure 17.1(a). Beginning in the start state, it must choose an action at each time step. The interaction with the environment terminates when the agent reaches one of the goal states, marked +1 or –1. In each location, the available actions are called *Up*, *Down*, *Left*, and *Right*. We will assume for now that the environment is **fully observable**, so that the agent always knows where it is.

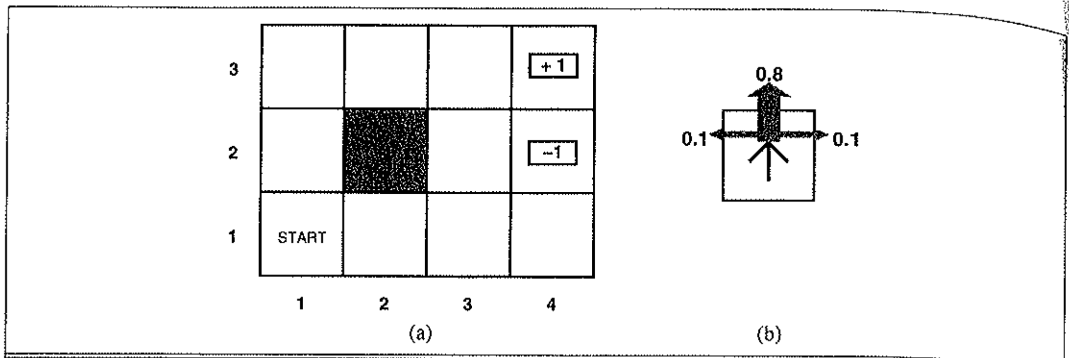

**Figure 17.1** (a) A simple $4 \times 3$ environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and –1, respectively, and all other states have a reward of –0.04.

If the environment were deterministic, a solution would be easy: [*Up, Up, Right, Right, Right*]. Unfortunately, the environment won't always go along with this solution, because the actions are unreliable. The particular model of stochastic motion that we adopt is illustrated in Figure 17.1(b). Each action achieves the intended effect with probability 0.8, but the rest of the time, the action moves the agent at right angles to the intended direction. Furthermore, if the agent bumps into a wall, it stays in the same square. For example, from the start square (1,1), the action *Up* moves the agent to (1,2) with probability 0.8, but with probability 0.1, it moves right to (2,1), and with probability 0.1, it moves left, bumps into the wall, and stays in (1,1). In such an environment, the sequence $[Up, Up, Right, Right, Right]$ goes up around the barrier and reaches the goal state at (4,3) with probability $0.8^5 = 0.32768$. There is also a small chance of accidentally reaching the goal by going the other way around with probability $0.1^4 \times 0.8$, for a grand total of 0.32776. (See also Exercise 17.1.)

TRANSITION MODEL

A specification of the outcome probabilities for each action in each possible state is called a **transition model** (or just "model," whenever no confusion can arise). We will use $T(s, a, s')$ to denote the probability of reaching state $s'$ if action $a$ is done in state $s$. We will assume that transitions are **Markovian** in the sense of Chapter 15, that is, the probability of reaching $s'$ from $s$ depends only on $s$ and not on the history of earlier states. For now, you can think of $T(s, a, s')$ as a big three-dimensional table containing probabilities. Later, in Section 17.5, we will see that the transition model can be represented as a **dynamic Bayesian network**, just as in Chapter 15.

REWARD

To complete the definition of the task environment, we must specify the utility function for the agent. Because the decision problem is sequential, the utility function will depend on a sequence of states—an **environment history**—rather than on a single state. Later in this section, we will investigate how such utility functions can be specified in general; for now, we will simply stipulate that in each state $s$, the agent receives a **reward** $R(s)$, which may be positive or negative, but must be bounded. For our particular example, the reward is $-0.04$ in all states except the terminal states (which have rewards +1 and –1). The utility of an environment history is just (for now) the *sum* of the rewards received. For example, if the agent reaches the +1 state after 10 steps, its total utility will be 0.6. The negative reward of $-0.04$ gives the agent an incentive to reach (4,3) quickly, so our environment is a stochastic generalization of the search problems of Chapter 3. Another way of saying this is that the agent does not enjoy living in this environment and so wants to get out of the game as soon as possible.

MARKOV DECISION PROCESS

The specification of a sequential decision problem for a fully observable environment with a Markovian transition model and additive rewards is called a **Markov decision process**, or **MDP**. An MDP is defined by the following three components:

Initial State: $S_0$
Transition Model: $T(s, a, s')$
Reward Function:[1] $R(s)$

The next question is, what does a solution to the problem look like? We have seen that any fixed action sequence won't solve the problem, because the agent might end up in a state other than the goal. Therefore, a solution must specify what the agent should do for *any* state that the agent might reach. A solution of this kind is called a **policy**. We usually denote a policy by $\pi$, and $\pi(s)$ is the action recommended by the policy $\pi$ for state $s$. If the agent has a complete policy, then no matter what the outcome of any action, the agent will always know what to do next.

POLICY

Each time a given policy is executed starting from the initial state, the stochastic nature of the environment will lead to a different environment history. The quality of a policy is therefore measured by the *expected* utility of the possible environment histories generated by that policy. An **optimal policy** is a policy that yields the highest expected utility. We use $\pi^*$ to denote an optimal policy. Given $\pi^*$, the agent decides what to do by consulting its current percept, which tells it the current state $s$, and then executing the action $\pi^*(s)$. A policy represents the agent function explicitly and is therefore a description of a simple reflex agent, computed from the information used for a utility-based agent.

OPTIMAL POLICY

An optimal policy for the world of Figure 17.1 is shown in Figure 17.2(a). Notice that, because the cost of taking a step is fairly small compared with the penalty for ending up in (4,2) by accident, the optimal policy for the state (3,1) is conservative. The policy recommends taking the long way round, rather than taking the short cut and thereby risking entering (4,2).

The balance of risk and reward changes depends on the value of $R(s)$ for the nonterminal states. Figure 17.2(b) shows optimal policies for four different ranges of $R(s)$. When $R(s) \leq -1.6284$, life is so painful that the agent heads straight for the nearest exit, even if the exit is worth $-1$. When $-0.4278 \leq R(s) \leq -0.0850$, life is quite unpleasant; the agent takes the shortest route to the +1 state and is willing to risk falling into the $-1$ state by accident. In particular, the agent takes the shortcut from (3,1). When life is only slightly dreary ($-0.0221 < R(s) < 0$), the optimal policy takes *no risks at all*. In (4,1) and (3,2), the agent

[1] Some definitions of MDPs allow the reward to depend on the action and outcome too, so the reward function is $R(s, a, s')$. This simplifies the description of some environments but does not change the problem in any fundamental way.

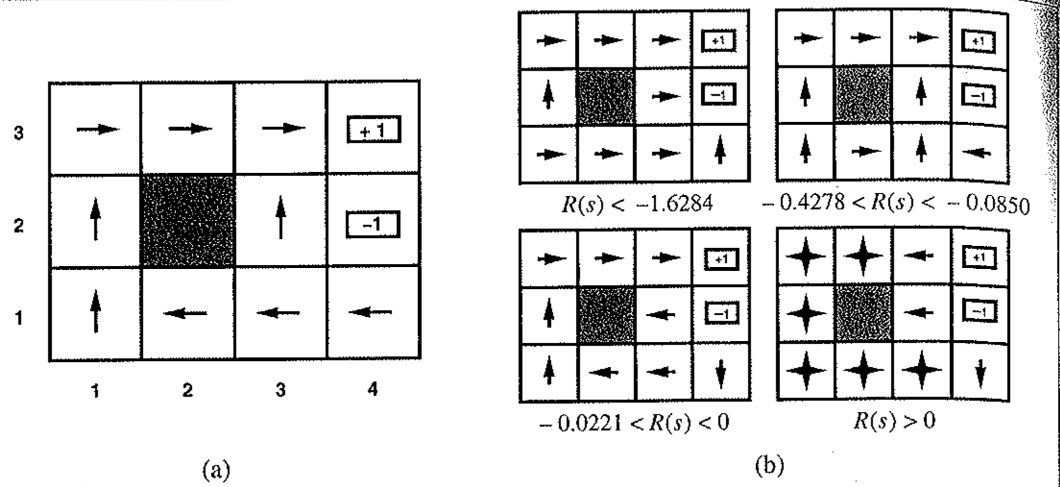

**Figure 17.2** (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

heads directly away from the $-1$ state so that it cannot fall in by accident, even though this means banging its head against the wall quite a few times. Finally, if $R(s) > 0$, then life is positively enjoyable and the agent avoids *both* exits. As long as the actions in (4,1), (3,2), and (3,3) are as shown, every policy is optimal, and the agent obtains infinite total reward because it never enters a terminal state. Surprisingly, it turns out that there are six other optimal policies for various ranges of $R(s)$; Exercise 17.7 asks you to find them.

The careful balancing of risk and reward is a characteristic of MDPs that does not arise in deterministic search problems; moreover, it is a characteristic of many real-world decision problems. For this reason, MDPs have been studied in several fields, including AI, operations research, economics, and control theory. Dozens of algorithms have been proposed for calculating optimal policies. In sections 17.2 and 17.3 we will describe two of the most important algorithm families. First, however, we must complete our investigation of utilities and policies for sequential decision problems.

## Optimality in sequential decision problems

In the MDP example in Figure 17.1, the performance of the agent was measured by a sum of rewards for the states visited. This choice of performance measure is not arbitrary, but it is not the only possibility. This section investigates the possible choices for the performance measure—that is, choices for the utility function on environment histories, which we will write as $U_h([s_0, s_1, \ldots, s_n])$. The section draws on ideas from Chapter 16 and is somewhat technical; the main points are summarized at the end.

FINITE HORIZON
INFINITE HORIZON

The first question to answer is whether there is a **finite horizon** or an **infinite horizon** for decision making. A finite horizon means that there is a *fixed* time $N$ after which nothing matters—the game is over, so to speak. Thus, $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$ for all $k > 0$. For example, suppose an agent starts at (3,1) in the $4 \times 3$ world of Figure 17.1,

NONSTATIONARY POLICY

STATIONARY POLICY

and suppose that $N = 3$. Then, to have any chance of reaching the +1 state, the agent must head directly for it, and the optimal action is to go *Up*. On the other hand, if $N = 100$ then there is plenty of time to take the safe route by going *Left*. *So, with a finite horizon, the optimal action in a given state could change over time.* We say that the optimal policy for a finite horizon is **nonstationary**. With no fixed time limit, on the other hand, there is no reason to behave differently in the same state at different times. Hence, the optimal action depends only on the current state, and the optimal policy is **stationary**. Policies for the infinite-horizon case are therefore simpler than those for the finite-horizon case, and we will deal mainly with the infinite-horizon case in this chapter.[2] Note that "infinite horizon" does not necessarily mean that all state sequences are infinite; it just means that there is no fixed deadline. In particular, there can be finite state sequences in an infinite-horizon MDP containing a terminal state.

STATIONARY PREFERENCE

The next question we must decide is how to calculate the utility of state sequences. We can view this as a question in **multiattribute utility theory** (see Section 16.4), where each state $s_i$ is viewed as an attribute of the state sequence $[s_0, s_1, s_2 \ldots]$. To obtain a simple expression in terms of the attributes, we will need to make some sort of preference independence assumption. The most natural assumption is that the agent's preferences between state sequences are **stationary**. Stationarity for preferences means the following: if two state sequences $[s_0, s_1, s_2, \ldots]$ and $[s'_0, s'_1, s'_2, \ldots]$ begin with the same state (i.e., $s_0 = s'_0$) then the two sequences should be preference-ordered the same way as the sequences $[s_1, s_2, \ldots]$ and $[s'_1, s'_2, \ldots]$. In English, this means that if you prefer one future to another starting tomorrow, then you should still prefer that future if it were to start today. Stationarity is a fairly innocuous-looking assumption with very strong consequences: it turns out that under stationarity there are just two ways to assign utilities to sequences:

ADDITIVE REWARDS

1. **Additive rewards**: The utility of a state sequence is
$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots .$$
The $4 \times 3$ world in Figure 17.1 uses additive rewards. Notice that additivity was used implicitly in our use of path cost functions in heuristic search algorithms (Chapter 4).

DISCOUNTED REWARDS

DISCOUNT FACTOR

2. **Discounted rewards**: The utility of a state sequence is
$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots ,$$
where the **discount factor** $\gamma$ is a number between 0 and 1. The discount factor describes the preference of an agent for current rewards over future rewards. When $\gamma$ is close to 0, rewards in the distant future are viewed as insignificant. When $\gamma$ is 1, discounted rewards are exactly equivalent to additive rewards, so additive rewards are a special case of discounted rewards. Discounting appears to be a good model of both animal and human preferences over time. A discount factor of $\gamma$ is equivalent to an interest rate of $(1/\gamma) - 1$.

For reasons that will shortly become clear, we will assume discounted rewards in the remainder of the chapter, although sometimes we will allow $\gamma = 1$.

[2] This is for completely observable environments. We will see later that for partially observable environments, the infinite-horizon case is not so simple.

Lurking beneath our choice of infinite horizons is a problem: if the environment does not contain a terminal state, or if the agent never reaches one, then all environment histories will be infinitely long, and utilities with additive rewards will generally be infinite. Now, we can agree that $+\infty$ is better than $-\infty$, but comparing two state sequences, both having $+\infty$ utility is more difficult. There are three solutions, two of which we have seen already:

1. With discounted rewards, the utility of an infinite sequence is *finite*. In fact, if rewards are bounded by $R_{\max}$ and $\gamma < 1$, we have
$$U_h([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = R_{\max}/(1-\gamma)\,, \qquad (17.1)$$
using the standard formula for the sum of an infinite geometric series.
2. If the environment contains terminal states *and if the agent is guaranteed to get to one eventually*, then we will never need to compare infinite sequences. A policy that is guaranteed to reach a terminal state is called a **proper policy**. With proper policies, we can use $\gamma = 1$ (i.e., additive rewards). The first three policies shown in Figure 17.2(b) are proper, but the fourth is improper. It gains infinite total reward by staying away from the terminal states when the reward for the nonterminal states is positive. The existence of improper policies can cause the standard algorithms for solving MDPs to fail with additive rewards, and so provides a good reason for using discounted rewards.

PROPER POLICY

AVERAGE REWARD

3. Another possibility is to compare infinite sequences in terms of the **average reward** obtained per time step. Suppose that square (1,1) in the $4 \times 3$ world has a reward of 0.1 while the other nonterminal states have a reward of 0.01. Then a policy that does its best to stay in (1,1) will have higher average reward than one that stays elsewhere. Average reward is a useful criterion for some problems, but the analysis of average-reward algorithms is beyond the scope of this book.

In sum, the use of discounted rewards presents the fewest difficulties in evaluating state sequences. The final step is to show how to choose between policies, bearing in mind that a given policy $\pi$ generates not one state sequence, but a whole range of possible state sequences, each with a specific probability determined by the transition model for the environment. Thus, the value of a policy is the *expected* sum of discounted rewards obtained, where the expectation is taken over all possible state sequences that could occur, given that the policy is executed. An optimal policy $\pi^*$ satisfies
$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]\,. \qquad (17.2)$$
The next two sections describe algorithms for finding optimal policies.