16 MAKING SIMPLE DECISIONS

In which we see how an agent should make decisions so that it gets what it wants—on average, at least.

In this chapter, we fill in the details of how utility theory combines with probability theory to yield a decision-theoretic agent—an agent that can make rational decisions based on what it believes and what it wants. Such an agent can make decisions in contexts in which uncertainty and conflicting goals leave a logical agent with no way to decide: a goal-based agent has a binary distinction between good (goal) and bad (non-goal) states, while a decision-theoretic agent has a continuous measure of outcome quality.

Section 16.1 introduces the basic principle of decision theory: the maximization of expected utility. Section 16.2 shows that the behavior of any rational agent can be captured by supposing a utility function that is being maximized. Section 16.3 discusses the nature of utility functions in more detail, and in particular their relation to individual quantities such as money. Section 16.4 shows how to handle utility functions that depend on several quantities. In Section 16.5, we describe the implementation of decision-making systems. In particular, we introduce a formalism called a **decision network** (also known as an **influence diagram**) that extends Bayesian networks by incorporating actions and utilities. The remainder of the chapter discusses issues that arise in applications of decision theory to expert systems.

16.1 COMBINING BELIEFS AND DESIRES UNDER UNCERTAINTY

Decision theory, in its simplest form, deals with choosing among actions based on the desirability of their *immediate* outcomes; that is, the environment is assumed to be episodic in the sense defined on page 43. (This assumption is relaxed in Chapter 17.) In Chapter 3 we used the notation $\text{RESULT}(s_0, a)$ for the state that is the deterministic outcome of taking action a in state s_0. In this chapter we deal with nondeterministic partially observable environments. Since the agent may not know the current state, we omit it and define $\text{RESULT}(a)$ as a *random variable* whose values are the possible outcome states. The probability of outcome s', given evidence observations $\mathbf{e}$, is written

$$P(\text{RESULT}(a) = s' \mid a, \mathbf{e}) \ ,$$

where the a on the right-hand side of the conditioning bar stands for the event that action a is executed.[1]

UTILITY FUNCTION

EXPECTED UTILITY

The agent's preferences are captured by a **utility function**, $U(s)$, which assigns a single number to express the desirability of a state. The **expected utility** of an action given the evidence, $EU(a|\mathbf{e})$, is just the average utility value of the outcomes, weighted by the probability that the outcome occurs:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s' \mid a, \mathbf{e})\, U(s') \ . \tag{16.1}$$

MAXIMUM EXPECTED UTILITY

The principle of **maximum expected utility** (MEU) says that a rational agent should choose the action that maximizes the agent's expected utility:

$$action = \operatorname*{argmax}_a EU(a|\mathbf{e})$$

In a sense, the MEU principle could be seen as defining all of AI. All an intelligent agent has to do is calculate the various quantities, maximize utility over its actions, and away it goes. But this does not mean that the AI problem is *solved* by the definition!

The MEU principle *formalizes* the general notion that the agent should "do the right thing," but goes only a small distance toward a full *operationalization* of that advice. Estimating the state of the world requires perception, learning, knowledge representation, and inference. Computing $P(\text{RESULT}(a) \mid a, \mathbf{e})$ requires a complete causal model of the world and, as we saw in Chapter 14, NP-hard inference in (very large) Bayesian networks. Computing the outcome utilities $U(s')$ often requires searching or planning, because an agent may not know how good a state is until it knows where it can get to from that state. So, decision theory is not a panacea that solves the AI problem—but it does provide a useful framework.

The MEU principle has a clear relation to the idea of performance measures introduced in Chapter 2. The basic idea is simple. Consider the environments that could lead to an agent having a given percept history, and consider the different agents that we could design.

If an agent acts so as to maximize a utility function that correctly reflects the performance measure, then the agent will achieve the highest possible performance score (averaged over all the possible environments). This is the central justification for the MEU principle itself. While the claim may seem tautological, it does in fact embody a very important transition from a global, external criterion of rationality—the performance measure over environment histories—to a local, internal criterion involving the maximization of a utility function applied to the next state.

16.2 THE BASIS OF UTILITY THEORY

Intuitively, the principle of Maximum Expected Utility (MEU) seems like a reasonable way to make decisions, but it is by no means obvious that it is the *only* rational way. After all, why should maximizing the *average* utility be so special? What's wrong with an agent that

[1] Classical decision theory leaves the current state S_0 implicit, but we could make it explicit by writing $P(\text{RESULT}(a) = s' \mid a, \mathbf{e}) = \sum_s P(\text{RESULT}(s, a) = s' \mid a) P(S_0 = s \mid \mathbf{e})$.

maximizes the weighted sum of the cubes of the possible utilities, or tries to minimize the worst possible loss? Could an agent act rationally just by expressing preferences between states, without giving them numeric values? Finally, why should a utility function with the required properties exist at all? We shall see.

16.2.1 Constraints on rational preferences

These questions can be answered by writing down some constraints on the preferences that a rational agent should have and then showing that the MEU principle can be derived from the constraints. We use the following notation to describe an agent's preferences:

$A \succ B$ the agent prefers A over B.
$A \sim B$ the agent is indifferent between A and B.
$A \succsim B$ the agent prefers A over B or is indifferent between them.

Now the obvious question is, what sorts of things are A and B? They could be states of the world, but more often than not there is uncertainty about what is really being offered. For example, an airline passenger who is offered "the pasta dish or the chicken" does not know what lurks beneath the tinfoil cover.[2] The pasta could be delicious or congealed, the chicken juicy or overcooked beyond recognition. We can think of the set of outcomes for each action as a **lottery**—think of each action as a ticket. A lottery L with possible outcomes $S_1, \ldots, S_n$ that occur with probabilities $p_1, \ldots, p_n$ is written

LOTTERY

$$L = [p_1, S_1;\ p_2, S_2;\ \ldots\ p_n, S_n]\ .$$

In general, each outcome S_i of a lottery can be either an atomic state or another lottery. The primary issue for utility theory is to understand how preferences between complex lotteries are related to preferences between the underlying states in those lotteries. To address this issue we list six constraints that we require any reasonable preference relation to obey:

ORDERABILITY

- **Orderability**: Given any two lotteries, a rational agent must either prefer one to the other or else rate the two as equally preferable. That is, the agent cannot avoid deciding. As we said on page 490, refusing to bet is like refusing to allow time to pass.

  $$\text{Exactly one of } (A \succ B),\ (B \succ A),\ \text{ or } (A \sim B) \text{ holds.}$$

TRANSITIVITY

- **Transitivity**: Given any three lotteries, if an agent prefers A to B and prefers B to C, then the agent must prefer A to C.

  $$(A \succ B) \wedge (B \succ C)\ \Rightarrow\ (A \succ C)\ .$$

CONTINUITY

- **Continuity**: If some lottery B is between A and C in preference, then there is some probability p for which the rational agent will be indifferent between getting B for sure and the lottery that yields A with probability p and C with probability $1 - p$.

  $$A \succ B \succ C\ \Rightarrow\ \exists p\ \ [p, A;\ 1-p, C] \sim B\ .$$

SUBSTITUTABILITY

- **Substitutability**: If an agent is indifferent between two lotteries A and B, then the agent is indifferent between two more complex lotteries that are the same except that B

[2] We apologize to readers whose local airlines no longer offer food on long flights.

is substituted for A in one of them. This holds regardless of the probabilities and the other outcome(s) in the lotteries.

$$A \sim B \;\Rightarrow\; [p, A;\; 1-p, C] \sim [p, B; 1-p, C] \,.$$

This also holds if we substitute $\succ$ for $\sim$ in this axiom.

MONOTONICITY

- **Monotonicity**: Suppose two lotteries have the same two possible outcomes, A and B. If an agent prefers A to B, then the agent must prefer the lottery that has a higher probability for A (and vice versa).

$$A \succ B \;\Rightarrow\; (p > q \;\Leftrightarrow\; [p, A;\; 1-p, B] \succ [q, A;\; 1-q, B]) \,.$$

DECOMPOSABILITY

- **Decomposability**: Compound lotteries can be reduced to simpler ones using the laws of probability. This has been called the "no fun in gambling" rule because it says that two consecutive lotteries can be compressed into a single equivalent lottery, as shown in Figure 16.1(b).[3]

$$[p, A;\; 1-p, [q, B;\; 1-q, C]] \sim [p, A;\; (1-p)q, B;\; (1-p)(1-q), C] \,.$$

These constraints are known as the axioms of utility theory. Each axiom can be motivated by showing that an agent that violates it will exhibit patently irrational behavior in some situations. For example, we can motivate transitivity by making an agent with nontransitive preferences give us all its money. Suppose that the agent has the nontransitive preferences $A \succ B \succ C \succ A$, where A, B, and C are goods that can be freely exchanged. If the agent currently has A, then we could offer to trade C for A plus one cent. The agent prefers C, and so would be willing to make this trade. We could then offer to trade B for C, extracting another cent, and finally trade A for B. This brings us back where we started from, except that the agent has given us three cents (Figure 16.1(a)). We can keep going around the cycle until the agent has no money at all. Clearly, the agent has acted irrationally in this case.

16.2.2 Preferences lead to utility

Notice that the axioms of utility theory are really axioms about preferences—they say nothing about a utility function. But in fact from the axioms of utility we can derive the following consequences (for the proof, see von Neumann and Morgenstern, 1944):

- **Existence of Utility Function**: If an agent's preferences obey the axioms of utility, then there exists a function U such that $U(A) > U(B)$ if and only if A is preferred to B, and $U(A) = U(B)$ if and only if the agent is indifferent between A and B.

$$U(A) > U(B) \;\Leftrightarrow\; A \succ B$$
$$U(A) = U(B) \;\Leftrightarrow\; A \sim B$$

- **Expected Utility of a Lottery**: The utility of a lottery is the sum of the probability of each outcome times the utility of that outcome.

$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i) \,.$$

[3] We can account for the enjoyment of gambling by encoding gambling events into the state description; for example, "Have $10 and gambled" could be preferred to "Have $10 and didn't gamble."

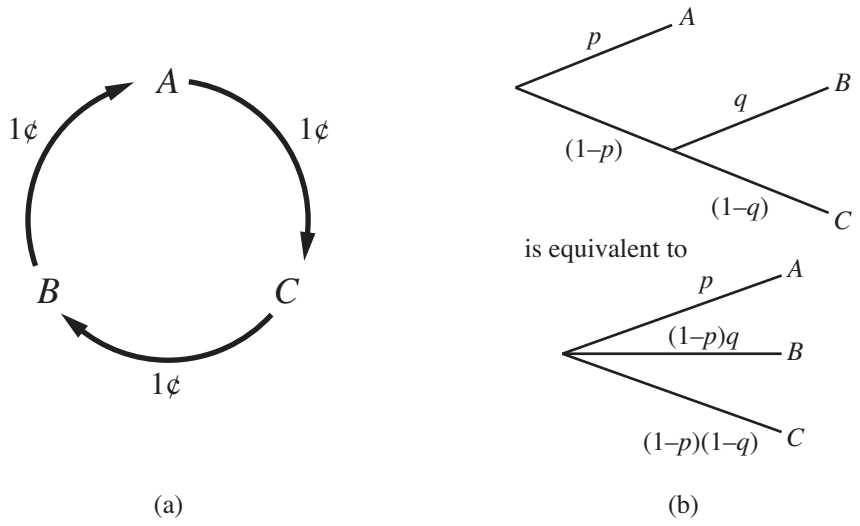


Figure 16.1 (a) A cycle of exchanges showing that the nontransitive preferences $A \succ B \succ C \succ A$ result in irrational behavior. (b) The decomposability axiom.

In other words, once the probabilities and utilities of the possible outcome states are specified, the utility of a compound lottery involving those states is completely determined. Because the outcome of a nondeterministic action is a lottery, it follows that an agent can act rationally—that is, consistently with its preferences—only by choosing an action that maximizes expected utility according to Equation (16.1).

The preceding theorems establish that a utility function *exists* for any rational agent, but they do not establish that it is *unique*. It is easy to see, in fact, that an agent's behavior would not change if its utility function $U(S)$ were transformed according to

$$U'(S) = aU(S) + b\,, \tag{16.2}$$

where a and b are constants and $a > 0$; an affine transformation.[4] This fact was noted in Chapter 5 for two-player games of chance; here, we see that it is completely general.

VALUE FUNCTION
ORDINAL UTILITY FUNCTION

As in game-playing, in a deterministic environment an agent just needs a preference ranking on states—the numbers don't matter. This is called a **value function** or **ordinal utility function**.

It is important to remember that the existence of a utility function that describes an agent's preference behavior does not necessarily mean that the agent is *explicitly* maximizing that utility function in its own deliberations. As we showed in Chapter 2, rational behavior can be generated in any number of ways. By observing a rational agent's preferences, however, an observer can construct the utility function that represents what the agent is actually trying to achieve (even if the agent doesn't know it).

[4] In this sense, utilities resemble temperatures: a temperature in Fahrenheit is 1.8 times the Celsius temperature plus 32. You get the same results in either measurement system.

16.3 UTILITY FUNCTIONS

Utility is a function that maps from lotteries to real numbers. We know there are some axioms on utilities that all rational agents must obey. Is that all we can say about utility functions? Strictly speaking, that is it: an agent can have any preferences it likes. For example, an agent might prefer to have a prime number of dollars in its bank account; in which case, if it had \$16 it would give away \$3. This might be unusual, but we can't call it irrational. An agent might prefer a dented 1973 Ford Pinto to a shiny new Mercedes. Preferences can also interact: for example, the agent might prefer prime numbers of dollars only when it owns the Pinto, but when it owns the Mercedes, it might prefer more dollars to fewer. Fortunately, the preferences of real agents are usually more systematic, and thus easier to deal with.

16.3.1 Utility assessment and utility scales

If we want to build a decision-theoretic system that helps the agent make decisions or acts on his or her behalf, we must first work out what the agent's utility function is. This process, often called **preference elicitation**, involves presenting choices to the agent and using the observed preferences to pin down the underlying utility function.

PREFERENCE ELICITATION

Equation (16.2) says that there is no absolute scale for utilities, but it is helpful, nonetheless, to establish *some* scale on which utilities can be recorded and compared for any particular problem. A scale can be established by fixing the utilities of any two particular outcomes, just as we fix a temperature scale by fixing the freezing point and boiling point of water. Typically, we fix the utility of a "best possible prize" at $U(S) = u_\top$ and a "worst possible catastrophe" at $U(S) = u_\bot$. **Normalized utilities** use a scale with $u_\bot = 0$ and $u_\top = 1$.

NORMALIZED UTILITIES

Given a utility scale between $u_\top$ and $u_\bot$, we can assess the utility of any particular prize S by asking the agent to choose between S and a **standard lottery** $[p, u_\top;\ (1-p), u_\bot]$. The probability p is adjusted until the agent is indifferent between S and the standard lottery. Assuming normalized utilities, the utility of S is given by p. Once this is done for each prize, the utilities for all lotteries involving those prizes are determined.

STANDARD LOTTERY

In medical, transportation, and environmental decision problems, among others, people's lives are at stake. In such cases, $u_\bot$ is the value assigned to immediate death (or perhaps many deaths). *Although nobody feels comfortable with putting a value on human life, it is a fact that tradeoffs are made all the time.* Aircraft are given a complete overhaul at intervals determined by trips and miles flown, rather than after every trip. Cars are manufactured in a way that trades off costs against accident survival rates. Paradoxically, a refusal to "put a monetary value on life" means that life is often *undervalued*. Ross Shachter relates an experience with a government agency that commissioned a study on removing asbestos from schools. The decision analysts performing the study assumed a particular dollar value for the life of a school-age child, and argued that the rational choice under that assumption was to remove the asbestos. The agency, morally outraged at the idea of setting the value of a life, rejected the report out of hand. It then decided against asbestos removal—implicitly asserting a lower value for the life of a child than that assigned by the analysts.

MICROMORT

Some attempts have been made to find out the value that people place on their own lives. One common "currency" used in medical and safety analysis is the **micromort**, a one in a million chance of death. If you ask people how much they would pay to avoid a risk—for example, to avoid playing Russian roulette with a million-barreled revolver—they will respond with very large numbers, perhaps tens of thousands of dollars, but their actual behavior reflects a much lower monetary value for a micromort. For example, driving in a car for 230 miles incurs a risk of one micromort; over the life of your car—say, 92,000 miles—that's 400 micromorts. People appear to be willing to pay about $10,000 (at 2009 prices) more for a safer car that halves the risk of death, or about $50 per micromort. A number of studies have confirmed a figure in this range across many individuals and risk types. Of course, this argument holds only for small risks. Most people won't agree to kill themselves for $50 million.

QALY

Another measure is the **QALY**, or quality-adjusted life year. Patients with a disability are willing to accept a shorter life expectancy to be restored to full health. For example, kidney patients on average are indifferent between living two years on a dialysis machine and one year at full health.

16.3.2 The utility of money

Utility theory has its roots in economics, and economics provides one obvious candidate for a utility measure: money (or more specifically, an agent's total net assets). The almost universal exchangeability of money for all kinds of goods and services suggests that money plays a significant role in human utility functions.

MONOTONIC PREFERENCE

It will usually be the case that an agent prefers more money to less, all other things being equal. We say that the agent exhibits a **monotonic preference** for more money. This does not mean that money behaves as a utility function, because it says nothing about preferences between *lotteries* involving money.

Suppose you have triumphed over the other competitors in a television game show. The host now offers you a choice: either you can take the $1,000,000 prize or you can gamble it on the flip of a coin. If the coin comes up heads, you end up with nothing, but if it comes up tails, you get $2,500,000. If you're like most people, you would decline the gamble and pocket the million. Are you being irrational?

EXPECTED MONETARY VALUE

Assuming the coin is fair, the **expected monetary value** (EMV) of the gamble is $\frac{1}{2}$($0) + $\frac{1}{2}$($2,500,000) = $1,250,000, which is more than the original $1,000,000. But that does not necessarily mean that accepting the gamble is a better decision. Suppose we use S_n to denote the state of possessing total wealth $\$n$, and that your current wealth is $\$k$. Then the expected utilities of the two actions of accepting and declining the gamble are

$$\begin{aligned} EU(Accept) &= \tfrac{1}{2}U(S_k) + \tfrac{1}{2}U(S_{k+2,500,000})\,, \\ EU(Decline) &= U(S_{k+1,000,000})\,. \end{aligned}$$

To determine what to do, we need to assign utilities to the outcome states. Utility is not directly proportional to monetary value, because the utility for your first million is very high (or so they say), whereas the utility for an additional million is smaller. Suppose you assign a utility of 5 to your current financial status (S_k), a 9 to the state $S_{k+2,500,000}$, and an 8 to the

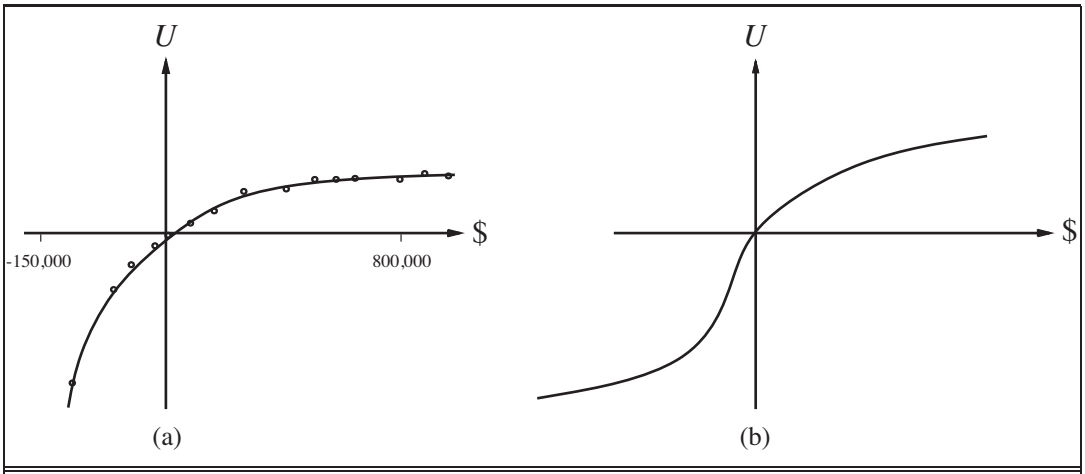


Figure 16.2 The utility of money. (a) Empirical data for Mr. Beard over a limited range. (b) A typical curve for the full range.

state $S_{k+1,000,000}$. Then the rational action would be to decline, because the expected utility of accepting is only 7 (less than the 8 for declining). On the other hand, a billionaire would most likely have a utility function that is locally linear over the range of a few million more, and thus would accept the gamble.

In a pioneering study of actual utility functions, Grayson (1960) found that the utility of money was almost exactly proportional to the *logarithm* of the amount. (This idea was first suggested by Bernoulli (1738); see Exercise 16.3.) One particular utility curve, for a certain Mr. Beard, is shown in Figure 16.2(a). The data obtained for Mr. Beard's preferences are consistent with a utility function

$$U(S_{k+n}) = -263.31 + 22.09 \log(n + 150,000)$$

for the range between $n = -\$150,000$ and $n = \$800,000$.

We should not assume that this is the definitive utility function for monetary value, but it is likely that most people have a utility function that is concave for positive wealth. Going into debt is bad, but preferences between different levels of debt can display a reversal of the concavity associated with positive wealth. For example, someone already $10,000,000 in debt might well accept a gamble on a fair coin with a gain of $10,000,000 for heads and a loss of $20,000,000 for tails.[5] This yields the S-shaped curve shown in Figure 16.2(b).

If we restrict our attention to the positive part of the curves, where the slope is decreasing, then for any lottery L, the utility of being faced with that lottery is less than the utility of being handed the expected monetary value of the lottery as a sure thing:

$$U(L) < U(S_{EMV(L)}) .$$

RISK-AVERSE

That is, agents with curves of this shape are **risk-averse**: they prefer a sure thing with a payoff that is less than the expected monetary value of a gamble. On the other hand, in the

RISK-SEEKING

"desperate" region at large negative wealth in Figure 16.2(b), the behavior is **risk-seeking**.

[5] Such behavior might be called desperate, but it is rational if one is already in a desperate situation.

CERTAINTY EQUIVALENT

The value an agent will accept in lieu of a lottery is called the **certainty equivalent** of the lottery. Studies have shown that most people will accept about $400 in lieu of a gamble that gives $1000 half the time and $0 the other half—that is, the certainty equivalent of the lottery is $400, while the EMV is $500. The difference between the EMV of a lottery and its certainty equivalent is called the **insurance premium**. Risk aversion is the basis for the insurance industry, because it means that insurance premiums are positive. People would rather pay a small insurance premium than gamble the price of their house against the chance of a fire. From the insurance company's point of view, the price of the house is very small compared with the firm's total reserves. This means that the insurer's utility curve is approximately linear over such a small region, and the gamble costs the company almost nothing.

INSURANCE PREMIUM

RISK-NEUTRAL

Notice that for *small* changes in wealth relative to the current wealth, almost any curve will be approximately linear. An agent that has a linear curve is said to be **risk-neutral**. For gambles with small sums, therefore, we expect risk neutrality. In a sense, this justifies the simplified procedure that proposed small gambles to assess probabilities and to justify the axioms of probability in Section 13.2.3.

16.3.3 Expected utility and post-decision disappointment

The rational way to choose the best action, a^*, is to maximize expected utility:

$$a^* = \operatorname*{argmax}_a EU(a|\mathbf{e}) \ .$$

If we have calculated the expected utility correctly according to our probability model, and if the probability model correctly reflects the underlying stochastic processes that generate the outcomes, then, on average, we will get the utility we expect if the whole process is repeated many times.

In reality, however, our model usually oversimplifies the real situation, either because we don't know enough (e.g., when making a complex investment decision) or because the computation of the true expected utility is too difficult (e.g., when estimating the utility of successor states of the root node in backgammon). In that case, we are really working with *estimates* $\widehat{EU}(a|\mathbf{e})$ of the true expected utility. We will assume, kindly perhaps, that the estimates are **unbiased**, that is, the expected value of the error, $E(\widehat{EU}(a|\mathbf{e}) - EU(a|\mathbf{e}))$, is zero. In that case, it still seems reasonable to choose the action with the highest estimated utility and to expect to receive that utility, on average, when the action is executed.

UNBIASED

Unfortunately, the real outcome will usually be significantly *worse* than we estimated, even though the estimate was unbiased! To see why, consider a decision problem in which there are k choices, each of which has true estimated utility of 0. Suppose that the error in each utility estimate has zero mean and standard deviation of 1, shown as the bold curve in Figure 16.3. Now, as we actually start to generate the estimates, some of the errors will be negative (pessimistic) and some will be positive (optimistic). Because we select the action with the *highest* utility estimate, we are obviously favoring the overly optimistic estimates, and that is the source of the bias. It is a straightforward matter to calculate the distribution of the maximum of the k estimates (see Exercise 16.11) and hence quantify the extent of our disappointment. The curve in Figure 16.3 for $k = 3$ has a mean around 0.85, so the average disappointment will be about 85% of the standard deviation in the utility estimates.

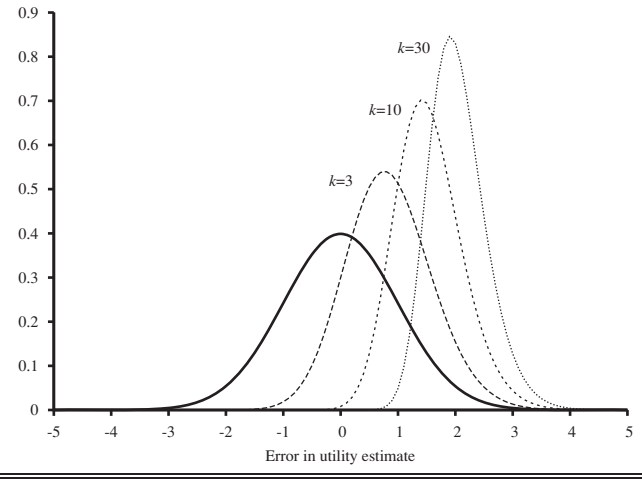


Figure 16.3 Plot of the error in each of k utility estimates and of the distribution of the maximum of k estimates for $k = 3$, 10, and 30.

With more choices, extremely optimistic estimates are more likely to arise: for $k = 30$, the disappointment will be around twice the standard deviation in the estimates.

OPTIMIZER'S CURSE

This tendency for the estimated expected utility of the best choice to be too high is called the **optimizer's curse** (Smith and Winkler, 2006). It afflicts even the most seasoned decision analysts and statisticians. Serious manifestations include believing that an exciting new drug that has cured 80% patients in a trial will cure 80% of patients (it's been chosen from $k =$ thousands of candidate drugs) or that a mutual fund advertised as having above-average returns will continue to have them (it's been chosen to appear in the advertisement out of $k =$ dozens of funds in the company's overall portfolio). It can even be the case that what appears to be the best choice may not be, if the variance in the utility estimate is high: a drug, selected from thousands tried, that has cured 9 of 10 patients is probably *worse* than one that has cured 800 of 1000.

The optimizer's curse crops up everywhere because of the ubiquity of utility-maximizing selection processes, so taking the utility estimates at face value is a bad idea. We can avoid the curse by using an explicit probability model $\mathbf{P}(\widehat{EU} \mid EU)$ of the error in the utility estimates. Given this model and a prior $\mathbf{P}(EU)$ on what we might reasonably expect the utilities to be, we treat the utility estimate, once obtained, as evidence and compute the posterior distribution for the true utility using Bayes' rule.

16.3.4 Human judgment and irrationality

NORMATIVE THEORY

DESCRIPTIVE THEORY

Decision theory is a **normative theory**: it describes how a rational agent *should* act. A **descriptive theory**, on the other hand, describes how actual agents—for example, humans—really do act. The application of economic theory would be greatly enhanced if the two coincided, but there appears to be some experimental evidence to the contrary. The evidence suggests that humans are "predictably irrational" (Ariely, 2009).

The best-known problem is the Allais paradox (Allais, 1953). People are given a choice between lotteries A and B and then between C and D, which have the following prizes:

A : 80% chance of \$4000 C : 20% chance of \$4000
B : 100% chance of \$3000 D : 25% chance of \$3000

Most people consistently prefer B over A (taking the sure thing), and C over D (taking the higher EMV). The normative analysis disagrees! We can see this most easily if we use the freedom implied by Equation (16.2) to set $U(\$0) = 0$. In that case, then $B \succ A$ implies that $U(\$3000) > 0.8\,U(\$4000)$, whereas $C \succ D$ implies exactly the reverse. In other words, there is no utility function that is consistent with these choices. One explanation for the apparently irrational preferences is the **certainty effect** (Kahneman and Tversky, 1979): people are strongly attracted to gains that are certain. There are several reasons why this may be so. First, people may prefer to reduce their computational burden; by choosing certain outcomes, they don't have to compute with probabilities. But the effect persists even when the computations involved are very easy ones. Second, people may distrust the legitimacy of the stated probabilities. I trust that a coin flip is roughly 50/50 if I have control over the coin and the flip, but I may distrust the result if the flip is done by someone with a vested interest in the outcome.[6] In the presence of distrust, it might be better to go for the sure thing.[7] Third, people may be accounting for their emotional state as well as their financial state. People know they would experience **regret** if they gave up a certain reward (B) for an 80% chance at a higher reward and then lost. In other words, if A is chosen, there is a 20% chance of getting no money *and feeling like a complete idiot*, which is worse than just getting no money. So perhaps people who choose B over A and C over D are not being irrational; they are just saying that they are willing to give up \$200 of EMV to avoid a 20% chance of feeling like an idiot.

CERTAINTY EFFECT

REGRET

A related problem is the Ellsberg paradox. Here the prizes are fixed, but the probabilities are underconstrained. Your payoff will depend on the color of a ball chosen from an urn. You are told that the urn contains 1/3 red balls, and 2/3 either black or yellow balls, but you don't know how many black and how many yellow. Again, you are asked whether you prefer lottery A or B; and then C or D:

A : \$100 for a red ball C : \$100 for a red or yellow ball
B : \$100 for a black ball D : \$100 for a black or yellow ball .

It should be clear that if you think there are more red than black balls then you should prefer A over B and C over D; if you think there are fewer red than black you should prefer the opposite. But it turns out that most people prefer A over B and also prefer D over C, even though there is no state of the world for which this is rational. It seems that people have **ambiguity aversion**: A gives you a 1/3 chance of winning, while B could be anywhere between 0 and 2/3. Similarly, D gives you a 2/3 chance, while C could be anywhere between 1/3 and 3/3. Most people elect the known probability rather than the unknown unknowns.

AMBIGUITY AVERSION

[6] For example, the mathematician/magician Persi Diaconis can make a coin flip come out the way he wants every time (Landhuis, 2004).

[7] Even the sure thing may not be certain. Despite cast-iron promises, we have not yet received that \$27,000,000 from the Nigerian bank account of a previously unknown deceased relative.

FRAMING EFFECT

Yet another problem is that the exact wording of a decision problem can have a big impact on the agent's choices; this is called the **framing effect**. Experiments show that people like a medical procedure that it is described as having a "90% survival rate" about twice as much as one described as having a "10% death rate," even though these two statements mean exactly the same thing. This discrepancy in judgment has been found in multiple experiments and is about the same whether the subjects were patients in a clinic, statistically sophisticated business school students, or experienced doctors.

People feel more comfortable making *relative* utility judgments rather than absolute ones. I may have little idea how much I might enjoy the various wines offered by a restaurant. The restaurant takes advantage of this by offering a $200 bottle that it knows nobody will buy, but which serves to skew upward the customer's estimate of the value of all wines and make the $55 bottle seem like a bargain. This is called the **anchoring effect**.

ANCHORING EFFECT

If human informants insist on contradictory preference judgments, there is nothing that automated agents can do to be consistent with them. Fortunately, preference judgments made by humans are often open to revision in the light of further consideration. Paradoxes like the Allais paradox are greatly reduced (but not eliminated) if the choices are explained better. In work at the Harvard Business School on assessing the utility of money, Keeney and Raiffa (1976, p. 210) found the following:

> Subjects tend to be too risk-averse in the small and therefore . . . the fitted utility functions exhibit unacceptably large risk premiums for lotteries with a large spread. . . . Most of the subjects, however, can reconcile their inconsistencies and feel that they have learned an important lesson about how they want to behave. As a consequence, some subjects cancel their automobile collision insurance and take out more term insurance on their lives.

EVOLUTIONARY PSYCHOLOGY

The evidence for human irrationality is also questioned by researchers in the field of **evolutionary psychology**, who point to the fact that our brain's decision-making mechanisms did not evolve to solve word problems with probabilities and prizes stated as decimal numbers. Let us grant, for the sake of argument, that the brain has built-in neural mechanism for computing with probabilities and utilities, or something functionally equivalent; if so, the required inputs would be obtained through accumulated experience of outcomes and rewards rather than through linguistic presentations of numerical values. It is far from obvious that we can directly access the brain's built-in neural mechanisms by presenting decision problems in linguistic/numerical form. The very fact that different wordings of the *same decision problem* elicit different choices suggests that the decision problem itself is not getting through. Spurred by this observation, psychologists have tried presenting problems in uncertain reasoning and decision making in "evolutionarily appropriate" forms; for example, instead of saying "90% survival rate," the experimenter might show 100 stick-figure animations of the operation, where the patient dies in 10 of them and survives in 90. (Boredom is a complicating factor in these experiments!) With decision problems posed in this way, people seem to be much closer to rational behavior than previously suspected.

16.4 MULTIATTRIBUTE UTILITY FUNCTIONS

Decision making in the field of public policy involves high stakes, in both money and lives. For example, in deciding what levels of harmful emissions to allow from a power plant, policy makers must weigh the prevention of death and disability against the benefit of the power and the economic burden of mitigating the emissions. Siting a new airport requires consideration of the disruption caused by construction; the cost of land; the distance from centers of population; the noise of flight operations; safety issues arising from local topography and weather conditions; and so on. Problems like these, in which outcomes are characterized by two or more attributes, are handled by **multiattribute utility theory**.

MULTIATTRIBUTE UTILITY THEORY

We will call the attributes $\mathbf{X} = X_1, \ldots, X_n$; a complete vector of assignments will be $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$, where each x_i is either a numeric value or a discrete value with an assumed ordering on values. We will assume that higher values of an attribute correspond to higher utilities, all other things being equal. For example, if we choose $AbsenceOfNoise$ as an attribute in the airport problem, then the greater its value, the better the solution.[8] We begin by examining cases in which decisions can be made *without* combining the attribute values into a single utility value. Then we look at cases in which the utilities of attribute combinations can be specified very concisely.

16.4.1 Dominance

Suppose that airport site S_1 costs less, generates less noise pollution, and is safer than site S_2. One would not hesitate to reject S_2. We then say that there is **strict dominance** of S_1 over S_2. In general, if an option is of lower value on all attributes than some other option, it need not be considered further. Strict dominance is often very useful in narrowing down the field of choices to the real contenders, although it seldom yields a unique choice. Figure 16.4(a) shows a schematic diagram for the two-attribute case.

STRICT DOMINANCE

That is fine for the deterministic case, in which the attribute values are known for sure. What about the general case, where the outcomes are uncertain? A direct analog of strict dominance can be constructed, where, despite the uncertainty, all possible concrete outcomes for S_1 strictly dominate all possible outcomes for S_2. (See Figure 16.4(b).) Of course, this will probably occur even less often than in the deterministic case.

Fortunately, there is a more useful generalization called **stochastic dominance**, which occurs very frequently in real problems. Stochastic dominance is easiest to understand in the context of a single attribute. Suppose we believe that the cost of siting the airport at S_1 is uniformly distributed between \$2.8 billion and \$4.8 billion and that the cost at S_2 is uniformly distributed between \$3 billion and \$5.2 billion. Figure 16.5(a) shows these distributions, with cost plotted as a negative value. Then, given only the information that utility decreases with

STOCHASTIC DOMINANCE

[8] In some cases, it may be necessary to subdivide the range of values so that utility varies monotonically within each range. For example, if the $RoomTemperature$ attribute has a utility peak at 70°F, we would split it into two attributes measuring the difference from the ideal, one colder and one hotter. Utility would then be monotonically increasing in each attribute.

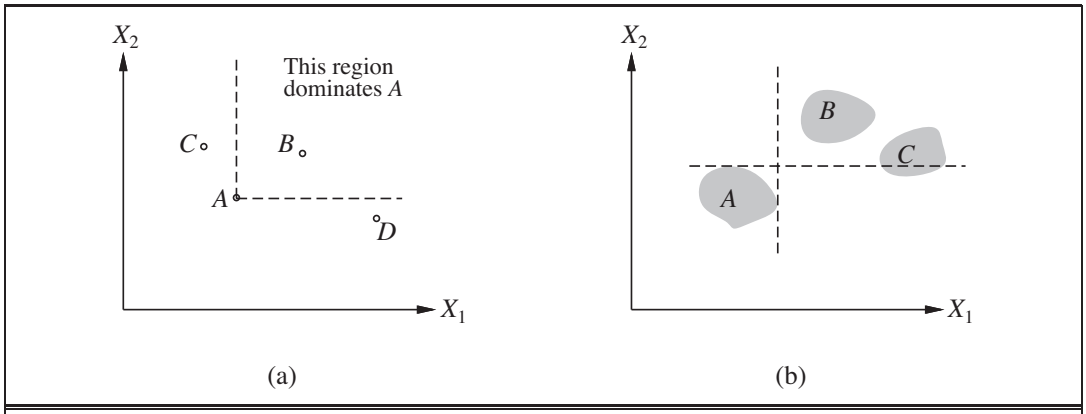


Figure 16.4 Strict dominance. (a) Deterministic: Option A is strictly dominated by B but not by C or D. (b) Uncertain: A is strictly dominated by B but not by C.

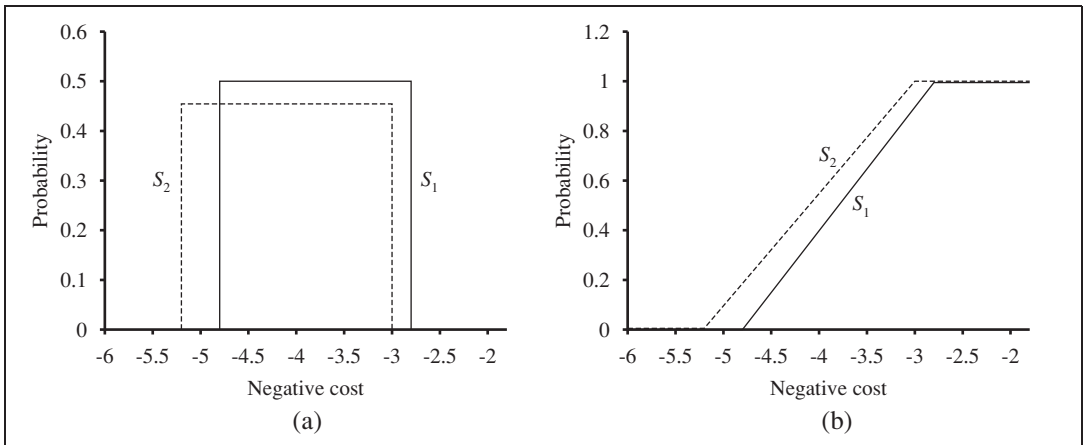


Figure 16.5 Stochastic dominance. (a) S_1 stochastically dominates S_2 on cost. (b) Cumulative distributions for the negative cost of S_1 and S_2.

cost, we can say that S_1 stochastically dominates S_2 (i.e., S_2 can be discarded). It is important to note that this does *not* follow from comparing the expected costs. For example, if we knew the cost of S_1 to be *exactly* \$3.8 billion, then we would be *unable* to make a decision without additional information on the utility of money. (It might seem odd that *more* information on the cost of S_1 could make the agent *less* able to decide. The paradox is resolved by noting that in the absence of exact cost information, the decision is easier to make but is more likely to be wrong.)

The exact relationship between the attribute distributions needed to establish stochastic dominance is best seen by examining the **cumulative distributions**, shown in Figure 16.5(b). (See also Appendix A.) The cumulative distribution measures the probability that the cost is less than or equal to any given amount—that is, it integrates the original distribution. If the cumulative distribution for S_1 is always to the right of the cumulative distribution for S_2,

then, stochastically speaking, S_1 is cheaper than S_2. Formally, if two actions A_1 and A_2 lead to probability distributions $p_1(x)$ and $p_2(x)$ on attribute X, then A_1 stochastically dominates A_2 on X if

$$\forall\, x \quad \int_{-\infty}^{x} p_1(x')\, dx' \leq \int_{-\infty}^{x} p_2(x')\, dx' \ .$$

The relevance of this definition to the selection of optimal decisions comes from the following property: *if A_1 stochastically dominates A_2, then for any monotonically nondecreasing utility function $U(x)$, the expected utility of A_1 is at least as high as the expected utility of A_2.* Hence, if an action is stochastically dominated by another action on all attributes, then it can be discarded.

The stochastic dominance condition might seem rather technical and perhaps not so easy to evaluate without extensive probability calculations. In fact, it can be decided very easily in many cases. Suppose, for example, that the construction transportation cost depends on the distance to the supplier. The cost itself is uncertain, but the greater the distance, the greater the cost. If S_1 is closer than S_2, then S_1 will dominate S_2 on cost. Although we will not present them here, there exist algorithms for propagating this kind of qualitative information among uncertain variables in **qualitative probabilistic networks**, enabling a system to make rational decisions based on stochastic dominance, without using any numeric values.

QUALITATIVE PROBABILISTIC NETWORKS

16.4.2 Preference structure and multiattribute utility

Suppose we have n attributes, each of which has d distinct possible values. To specify the complete utility function $U(x_1, \ldots, x_n)$, we need d^n values in the worst case. Now, the worst case corresponds to a situation in which the agent's preferences have no regularity at all. Multiattribute utility theory is based on the supposition that the preferences of typical agents have much more structure than that. The basic approach is to identify regularities in the preference behavior we would expect to see and to use what are called **representation theorems** to show that an agent with a certain kind of preference structure has a utility function

REPRESENTATION THEOREM

$$U(x_1, \ldots, x_n) = F[f_1(x_1), \ldots, f_n(x_n)] \ ,$$

where F is, we hope, a simple function such as addition. Notice the similarity to the use of Bayesian networks to decompose the joint probability of several random variables.

Preferences without uncertainty

Let us begin with the deterministic case. Remember that for deterministic environments the agent has a value function $V(x_1, \ldots, x_n)$; the aim is to represent this function concisely. The basic regularity that arises in deterministic preference structures is called **preference independence**. Two attributes X_1 and X_2 are preferentially independent of a third attribute X_3 if the preference between outcomes $\langle x_1, x_2, x_3\rangle$ and $\langle x'_1, x'_2, x_3\rangle$ does not depend on the particular value x_3 for attribute X_3.

PREFERENCE INDEPENDENCE

Going back to the airport example, where we have (among other attributes) $Noise$, $Cost$, and $Deaths$ to consider, one may propose that $Noise$ and $Cost$ are preferentially inde-

pendent of $Deaths$. For example, if we prefer a state with 20,000 people residing in the flight path and a construction cost of \$4 billion over a state with 70,000 people residing in the flight path and a cost of \$3.7 billion when the safety level is 0.06 deaths per million passenger miles in both cases, then we would have the same preference when the safety level is 0.12 or 0.03; and the same independence would hold for preferences between any other pair of values for $Noise$ and $Cost$. It is also apparent that $Cost$ and $Deaths$ are preferentially independent of $Noise$ and that $Noise$ and $Deaths$ are preferentially independent of $Cost$. We say that the set of attributes $\{Noise, Cost, Deaths\}$ exhibits **mutual preferential independence** (MPI). MPI says that, whereas each attribute may be important, it does not affect the way in which one trades off the other attributes against each other.

MUTUAL PREFERENTIAL INDEPENDENCE

Mutual preferential independence is something of a mouthful, but thanks to a remarkable theorem due to the economist Gérard Debreu (1960), we can derive from it a very simple form for the agent's value function: *If attributes $X_1, \ldots, X_n$ are mutually preferentially independent, then the agent's preference behavior can be described as maximizing the function*

$$V(x_1, \ldots, x_n) = \sum_i V_i(x_i) \ ,$$

where each V_i is a value function referring only to the attribute X_i. For example, it might well be the case that the airport decision can be made using a value function

$$V(noise, cost, deaths) = -noise \times 10^4 - cost - deaths \times 10^{12} \ .$$

ADDITIVE VALUE FUNCTION

A value function of this type is called an **additive value function**. Additive functions are an extremely natural way to describe an agent's preferences and are valid in many real-world situations. For n attributes, assessing an additive value function requires assessing n separate one-dimensional value functions rather than one n-dimensional function; typically, this represents an exponential reduction in the number of preference experiments that are needed. Even when MPI does not strictly hold, as might be the case at extreme values of the attributes, an additive value function might still provide a good approximation to the agent's preferences. This is especially true when the violations of MPI occur in portions of the attribute ranges that are unlikely to occur in practice.

To understand MPI better, it helps to look at cases where it *doesn't* hold. Suppose you are at a medieval market, considering the purchase of some hunting dogs, some chickens, and some wicker cages for the chickens. The hunting dogs are very valuable, but if you don't have enough cages for the chickens, the dogs will eat the chickens; hence, the tradeoff between dogs and chickens depends strongly on the number of cages, and MPI is violated. The existence of these kinds of interactions among various attributes makes it much harder to assess the overall value function.

Preferences with uncertainty

When uncertainty is present in the domain, we also need to consider the structure of preferences between lotteries and to understand the resulting properties of utility functions, rather than just value functions. The mathematics of this problem can become quite complicated, so we present just one of the main results to give a flavor of what can be done. The reader is referred to Keeney and Raiffa (1976) for a thorough survey of the field.

UTILITY INDEPENDENCE

The basic notion of **utility independence** extends preference independence to cover lotteries: a set of attributes **X** is utility independent of a set of attributes **Y** if preferences between lotteries on the attributes in **X** are independent of the particular values of the attributes in **Y**. A set of attributes is **mutually utility independent** (MUI) if each of its subsets is utility-independent of the remaining attributes. Again, it seems reasonable to propose that the airport attributes are MUI.

MUTUALLY UTILITY INDEPENDENT

MULTIPLICATIVE UTILITY FUNCTION

MUI implies that the agent's behavior can be described using a **multiplicative utility function** (Keeney, 1974). The general form of a multiplicative utility function is best seen by looking at the case for three attributes. For conciseness, we use U_i to mean $U_i(x_i)$:

$$U = k_1U_1 + k_2U_2 + k_3U_3 + k_1k_2U_1U_2 + k_2k_3U_2U_3 + k_3k_1U_3U_1 + k_1k_2k_3U_1U_2U_3 \ .$$

Although this does not look very simple, it contains just three single-attribute utility functions and three constants. In general, an n-attribute problem exhibiting MUI can be modeled using n single-attribute utilities and n constants. Each of the single-attribute utility functions can be developed independently of the other attributes, and this combination will be guaranteed to generate the correct overall preferences. Additional assumptions are required to obtain a purely additive utility function.

16.5 DECISION NETWORKS

In this section, we look at a general mechanism for making rational decisions. The notation is often called an **influence diagram** (Howard and Matheson, 1984), but we will use the more descriptive term **decision network**. Decision networks combine Bayesian networks with additional node types for actions and utilities. We use airport siting as an example.

INFLUENCE DIAGRAM

DECISION NETWORK

16.5.1 Representing a decision problem with a decision network

In its most general form, a decision network represents information about the agent's current state, its possible actions, the state that will result from the agent's action, and the utility of that state. It therefore provides a substrate for implementing utility-based agents of the type first introduced in Section 2.4.5. Figure 16.6 shows a decision network for the airport siting problem. It illustrates the three types of nodes used:

CHANCE NODES

- **Chance nodes** (ovals) represent random variables, just as they do in Bayesian networks. The agent could be uncertain about the construction cost, the level of air traffic and the potential for litigation, and the $Deaths$, $Noise$, and total $Cost$ variables, each of which also depends on the site chosen. Each chance node has associated with it a conditional distribution that is indexed by the state of the parent nodes. In decision networks, the parent nodes can include decision nodes as well as chance nodes. Note that each of the current-state chance nodes could be part of a large Bayesian network for assessing construction costs, air traffic levels, or litigation potentials.

DECISION NODES

- **Decision nodes** (rectangles) represent points where the decision maker has a choice of

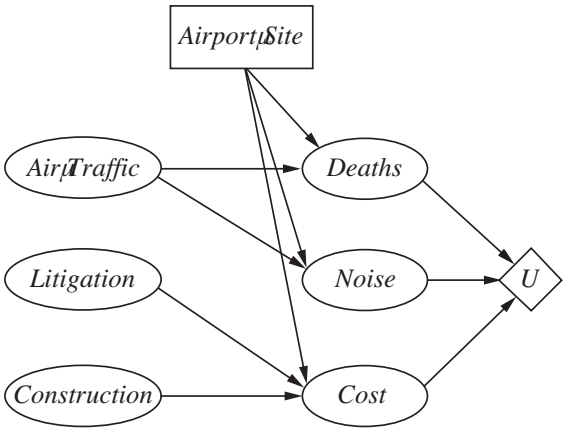


Figure 16.6 A simple decision network for the airport-siting problem.

actions. In this case, the $AirportSite$ action can take on a different value for each site under consideration. The choice influences the cost, safety, and noise that will result. In this chapter, we assume that we are dealing with a single decision node. Chapter 17 deals with cases in which more than one decision must be made.

UTILITY NODES

- **Utility nodes** (diamonds) represent the agent's utility function.[9] The utility node has as parents all variables describing the outcome that directly affect utility. Associated with the utility node is a description of the agent's utility as a function of the parent attributes. The description could be just a tabulation of the function, or it might be a parameterized additive or linear function of the attribute values.

A simplified form is also used in many cases. The notation remains identical, but the chance nodes describing the outcome state are omitted. Instead, the utility node is connected directly to the current-state nodes and the decision node. In this case, rather than representing a utility function on outcome states, the utility node represents the *expected* utility associated with each action, as defined in Equation (16.1) on page 611; that is, the node is associated with an **action-utility function** (also known as a **Q-function** in reinforcement learning, as described in Chapter 21). Figure 16.7 shows the action-utility representation of the airport siting problem.

ACTION-UTILITY FUNCTION

Notice that, because the $Noise$, $Deaths$, and $Cost$ chance nodes in Figure 16.6 refer to future states, they can never have their values set as evidence variables. Thus, the simplified version that omits these nodes can be used whenever the more general form can be used. Although the simplified form contains fewer nodes, the omission of an explicit description of the outcome of the siting decision means that it is less flexible with respect to changes in circumstances. For example, in Figure 16.6, a change in aircraft noise levels can be reflected by a change in the conditional probability table associated with the $Noise$ node, whereas a change in the weight accorded to noise pollution in the utility function can be reflected by

[9] These nodes are also called **value nodes** in the literature.

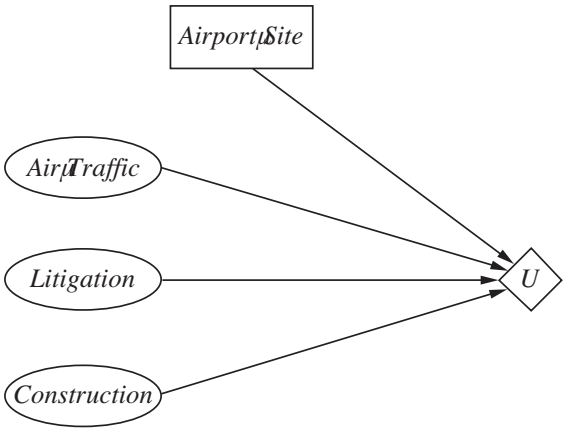


Figure 16.7 A simplified representation of the airport-siting problem. Chance nodes corresponding to outcome states have been factored out.

a change in the utility table. In the action-utility diagram, Figure 16.7, on the other hand, all such changes have to be reflected by changes to the action-utility table. Essentially, the action-utility formulation is a *compiled* version of the original formulation.

16.5.2 Evaluating decision networks

Actions are selected by evaluating the decision network for each possible setting of the decision node. Once the decision node is set, it behaves exactly like a chance node that has been set as an evidence variable. The algorithm for evaluating decision networks is the following:

1. Set the evidence variables for the current state.
2. For each possible value of the decision node:
   (a) Set the decision node to that value.
   (b) Calculate the posterior probabilities for the parent nodes of the utility node, using a standard probabilistic inference algorithm.
   (c) Calculate the resulting utility for the action.
3. Return the action with the highest utility.

This is a straightforward extension of the Bayesian network algorithm and can be incorporated directly into the agent design given in Figure 13.1 on page 484. We will see in Chapter 17 that the possibility of executing several actions in sequence makes the problem much more interesting.

16.6 THE VALUE OF INFORMATION

In the preceding analysis, we have assumed that all relevant information, or at least all available information, is provided to the agent before it makes its decision. In practice, this is

hardly ever the case. *One of the most important parts of decision making is knowing what questions to ask.* For example, a doctor cannot expect to be provided with the results of *all possible* diagnostic tests and questions at the time a patient first enters the consulting room.[10] Tests are often expensive and sometimes hazardous (both directly and because of associated delays). Their importance depends on two factors: whether the test results would lead to a significantly better treatment plan, and how likely the various test results are.

INFORMATION VALUE THEORY

This section describes **information value theory**, which enables an agent to choose what information to acquire. We assume that, prior to selecting a "real" action represented by the decision node, the agent can acquire the value of any of the potentially observable chance variables in the model. Thus, information value theory involves a simplified form of sequential decision making—simplified because the observation actions affect only the agent's **belief state**, not the external physical state. The value of any particular observation must derive from the potential to affect the agent's eventual physical action; and this potential can be estimated directly from the decision model itself.

16.6.1 A simple example

Suppose an oil company is hoping to buy one of n indistinguishable blocks of ocean-drilling rights. Let us assume further that exactly one of the blocks contains oil worth C dollars, while the others are worthless. The asking price of each block is C/n dollars. If the company is risk-neutral, then it will be indifferent between buying a block and not buying one.

Now suppose that a seismologist offers the company the results of a survey of block number 3, which indicates definitively whether the block contains oil. How much should the company be willing to pay for the information? The way to answer this question is to examine what the company would do if it had the information:

- With probability $1/n$, the survey will indicate oil in block 3. In this case, the company will buy block 3 for C/n dollars and make a profit of $C - C/n = (n-1)C/n$ dollars.
- With probability $(n-1)/n$, the survey will show that the block contains no oil, in which case the company will buy a different block. Now the probability of finding oil in one of the other blocks changes from $1/n$ to $1/(n-1)$, so the company makes an expected profit of $C/(n-1) - C/n = C/n(n-1)$ dollars.

Now we can calculate the expected profit, given the survey information:

$$\frac{1}{n} \times \frac{(n-1)C}{n} + \frac{n-1}{n} \times \frac{C}{n(n-1)} = C/n \; .$$

Therefore, the company should be willing to pay the seismologist up to C/n dollars for the information: the information is worth as much as the block itself.

The value of information derives from the fact that *with* the information, one's course of action can be changed to suit the *actual* situation. One can discriminate according to the situation, whereas without the information, one has to do what's best on average over the possible situations. In general, the value of a given piece of information is defined to be the difference in expected value between best actions before and after information is obtained.

[10] In the United States, the only question that is always asked beforehand is whether the patient has insurance.

16.6.2 A general formula for perfect information

VALUE OF PERFECT INFORMATION

It is simple to derive a general mathematical formula for the value of information. We assume that exact evidence can be obtained about the value of some random variable E_j (that is, we learn $E_j = e_j$), so the phrase **value of perfect information** (VPI) is used.[11]

Let the agent's initial evidence be $\mathbf{e}$. Then the value of the current best action α is defined by

$$EU(\alpha|\mathbf{e}) = \max_a \sum_{s'} P(\text{RESULT}(a) = s' \mid a, \mathbf{e})\, U(s') \ ,$$

and the value of the new best action (after the new evidence $E_j = e_j$ is obtained) will be

$$EU(\alpha_{e_j}|\mathbf{e}, e_j) = \max_a \sum_{s'} P(\text{RESULT}(a) = s' \mid a, \mathbf{e}, e_j)\, U(s') \ .$$

But E_j is a random variable whose value is *currently* unknown, so to determine the value of discovering E_j, given current information $\mathbf{e}$ we must average over all possible values e_{jk} that we might discover for E_j, using our *current* beliefs about its value:

$$VPI_{\mathbf{e}}(E_j) = \left(\sum_k P(E_j = e_{jk}|\mathbf{e})\ EU(\alpha_{e_{jk}}|\mathbf{e}, E_j = e_{jk}) \right) - EU(\alpha|\mathbf{e}) \ .$$

To get some intuition for this formula, consider the simple case where there are only two actions, a_1 and a_2, from which to choose. Their current expected utilities are U_1 and U_2. The information $E_j = e_{jk}$ will yield some new expected utilities U_1' and U_2' for the actions, but before we obtain E_j, we will have some probability distributions over the possible values of U_1' and U_2' (which we assume are independent).

Suppose that a_1 and a_2 represent two different routes through a mountain range in winter. a_1 is a nice, straight highway through a low pass, and a_2 is a winding dirt road over the top. Just given this information, a_1 is clearly preferable, because it is quite possible that a_2 is blocked by avalanches, whereas it is unlikely that anything blocks a_1. U_1 is therefore clearly higher than U_2. It is possible to obtain satellite reports E_j on the actual state of each road that would give new expectations, U_1' and U_2', for the two crossings. The distributions for these expectations are shown in Figure 16.8(a). Obviously, in this case, it is not worth the expense of obtaining satellite reports, because it is unlikely that the information derived from them will change the plan. With no change, information has no value.

Now suppose that we are choosing between two different winding dirt roads of slightly different lengths and we are carrying a seriously injured passenger. Then, even when U_1 and U_2 are quite close, the distributions of U_1' and U_2' are very broad. There is a significant possibility that the second route will turn out to be clear while the first is blocked, and in this

[11] There is no loss of expressiveness in requiring perfect information. Suppose we wanted to model the case in which we become somewhat more certain about a variable. We can do that by introducing *another* variable about which we learn perfect information. For example, suppose we initially have broad uncertainty about the variable $Temperature$. Then we gain the perfect knowledge $Thermometer = 37$; this gives us imperfect information about the true $Temperature$, and the uncertainty due to measurement error is encoded in the sensor model $\mathbf{P}(Thermometer \mid Temperature)$. See Exercise 16.17 for another example.

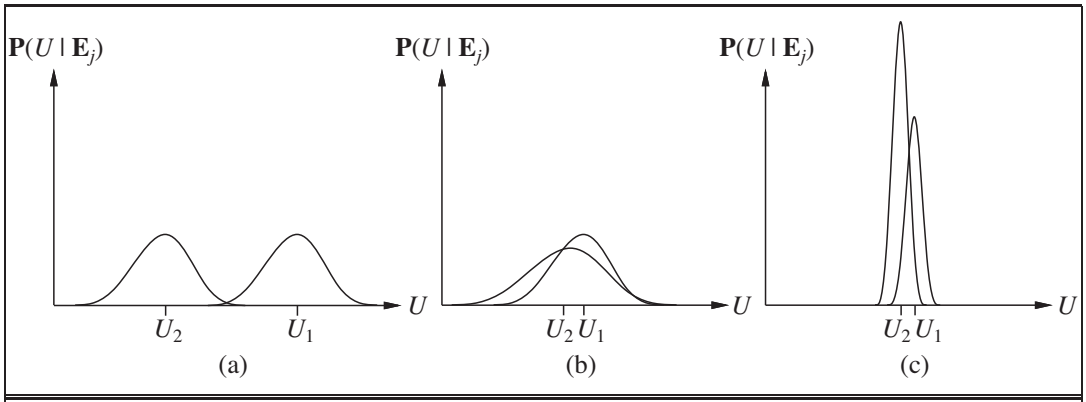


Figure 16.8 Three generic cases for the value of information. In (a), a_1 will almost certainly remain superior to a_2, so the information is not needed. In (b), the choice is unclear and the information is crucial. In (c), the choice is unclear, but because it makes little difference, the information is less valuable. (Note: The fact that U_2 has a high peak in (c) means that its expected value is known with higher certainty than U_1.)

case the difference in utilities will be very high. The VPI formula indicates that it might be worthwhile getting the satellite reports. Such a situation is shown in Figure 16.8(b).

Finally, suppose that we are choosing between the two dirt roads in summertime, when blockage by avalanches is unlikely. In this case, satellite reports might show one route to be more scenic than the other because of flowering alpine meadows, or perhaps wetter because of errant streams. It is therefore quite likely that we would change our plan if we had the information. In this case, however, the difference in value between the two routes is still likely to be very small, so we will not bother to obtain the reports. This situation is shown in Figure 16.8(c).

In sum, *information has value to the extent that it is likely to cause a change of plan and to the extent that the new plan will be significantly better than the old plan.*

16.6.3 Properties of the value of information

One might ask whether it is possible for information to be deleterious: can it actually have negative expected value? Intuitively, one should expect this to be impossible. After all, one could in the worst case just ignore the information and pretend that one has never received it. This is confirmed by the following theorem, which applies to any decision-theoretic agent:

The expected value of information is nonnegative:

$$\forall\, \mathbf{e}, E_j \quad VPI_{\mathbf{e}}(E_j) \geq 0\ .$$

The theorem follows directly from the definition of VPI, and we leave the proof as an exercise (Exercise 16.18). It is, of course, a theorem about *expected* value, not *actual* value. Additional information can easily lead to a plan that *turns out to* be worse than the original plan if the information happens to be misleading. For example, a medical test that gives a false positive result may lead to unnecessary surgery; but that does not mean that the test shouldn't be done.

It is important to remember that VPI depends on the current state of information, which is why it is subscripted. It can change as more information is acquired. For any given piece of evidence E_j, the value of acquiring it can go down (e.g., if another variable strongly constrains the posterior for E_j) or up (e.g., if another variable provides a clue on which E_j builds, enabling a new and better plan to be devised). Thus, VPI is not additive. That is,

$$VPI_{\mathbf{e}}(E_j, E_k) \neq VPI_{\mathbf{e}}(E_j) + VPI_{\mathbf{e}}(E_k) \qquad \text{(in general)} .$$

VPI is, however, order independent. That is,

$$VPI_{\mathbf{e}}(E_j, E_k) = VPI_{\mathbf{e}}(E_j) + VPI_{\mathbf{e},e_j}(E_k) = VPI_{\mathbf{e}}(E_k) + VPI_{\mathbf{e},e_k}(E_j) .$$

Order independence distinguishes sensing actions from ordinary actions and simplifies the problem of calculating the value of a sequence of sensing actions.

16.6.4 Implementation of an information-gathering agent

A sensible agent should ask questions in a reasonable order, should avoid asking questions that are irrelevant, should take into account the importance of each piece of information in relation to its cost, and should stop asking questions when that is appropriate. All of these capabilities can be achieved by using the value of information as a guide.

Figure 16.9 shows the overall design of an agent that can gather information intelligently before acting. For now, we assume that with each observable evidence variable E_j, there is an associated cost, $Cost(E_j)$, which reflects the cost of obtaining the evidence through tests, consultants, questions, or whatever. The agent requests what appears to be the most efficient observation in terms of utility gain per unit cost. We assume that the result of the action $Request(E_j)$ is that the next percept provides the value of E_j. If no observation is worth its cost, the agent selects a "real" action.

MYOPIC

The agent algorithm we have described implements a form of information gathering that is called **myopic**. This is because it uses the VPI formula shortsightedly, calculating the value of information as if only a single evidence variable will be acquired. Myopic control is based on the same heuristic idea as greedy search and often works well in practice. (For example, it has been shown to outperform expert physicians in selecting diagnostic tests.)

function INFORMATION-GATHERING-AGENT(*percept*) **returns** an *action*
  persistent: D, a decision network

  integrate *percept* into D
  $j \leftarrow$ the value that maximizes $VPI(E_j) \,/\, Cost(E_j)$
  if $VPI(E_j) > Cost(E_j)$
    return REQUEST(E_j)
  else return the best action from D

Figure 16.9 Design of a simple information-gathering agent. The agent works by repeatedly selecting the observation with the highest information value, until the cost of the next observation is greater than its expected benefit.

However, if there is no single evidence variable that will help a lot, a myopic agent might hastily take an action when it would have been better to request two or more variables first and then take action. A better approach in this situation would be to construct a *conditional plan* (as described in Section 11.3.2) that asks for variable values and takes different next steps depending on the answer.

One final consideration is the effect a series of questions will have on a human respondent. People may respond better to a series of questions if they "make sense," so some expert systems are built to take this into account, asking questions in an order that maximizes the total utility of the system and human rather than an order that maximizes value of information.

16.7 DECISION-THEORETIC EXPERT SYSTEMS

DECISION ANALYSIS

The field of **decision analysis**, which evolved in the 1950s and 1960s, studies the application of decision theory to actual decision problems. It is used to help make rational decisions in important domains where the stakes are high, such as business, government, law, military strategy, medical diagnosis and public health, engineering design, and resource management. The process involves a careful study of the possible actions and outcomes, as well as the preferences placed on each outcome. It is traditional in decision analysis to talk about two

DECISION MAKER

roles: the **decision maker** states preferences between outcomes, and the **decision analyst**

DECISION ANALYST

enumerates the possible actions and outcomes and elicits preferences from the decision maker to determine the best course of action. Until the early 1980s, the main purpose of decision analysis was to help humans make decisions that actually reflect their own preferences. As more and more decision processes become automated, decision analysis is increasingly used to ensure that the automated processes are behaving as desired.

Early expert system research concentrated on answering questions, rather than on making decisions. Those systems that did recommend actions rather than providing opinions on matters of fact generally did so using condition-action rules, rather than with explicit representations of outcomes and preferences. The emergence of Bayesian networks in the late 1980s made it possible to build large-scale systems that generated sound probabilistic inferences from evidence. The addition of decision networks means that expert systems can be developed that recommend optimal decisions, reflecting the preferences of the agent as well as the available evidence.

A system that incorporates utilities can avoid one of the most common pitfalls associated with the consultation process: confusing likelihood and importance. A common strategy in early medical expert systems, for example, was to rank possible diagnoses in order of likelihood and report the most likely. Unfortunately, this can be disastrous! For the majority of patients in general practice, the two most *likely* diagnoses are usually "There's nothing wrong with you" and "You have a bad cold," but if the third most likely diagnosis for a given patient is lung cancer, that's a serious matter. Obviously, a testing or treatment plan should depend both on probabilities and utilities. Current medical expert systems can take into account the value of information to recommend tests, and then describe a differential diagnosis.

We now describe the knowledge engineering process for decision-theoretic expert systems. As an example we consider the problem of selecting a medical treatment for a kind of congenital heart disease in children (see Lucas, 1996).

AORTIC COARCTATION

About 0.8% of children are born with a heart anomaly, the most common being **aortic coarctation** (a constriction of the aorta). It can be treated with surgery, angioplasty (expanding the aorta with a balloon placed inside the artery), or medication. The problem is to decide what treatment to use and when to do it: the younger the infant, the greater the risks of certain treatments, but one mustn't wait too long. A decision-theoretic expert system for this problem can be created by a team consisting of at least one domain expert (a pediatric cardiologist) and one knowledge engineer. The process can be broken down into the following steps:

Create a causal model. Determine the possible symptoms, disorders, treatments, and outcomes. Then draw arcs between them, indicating what disorders cause what symptoms, and what treatments alleviate what disorders. Some of this will be well known to the domain expert, and some will come from the literature. Often the model will match well with the informal graphical descriptions given in medical textbooks.

Simplify to a qualitative decision model. Since we are using the model to make treatment decisions and not for other purposes (such as determining the joint probability of certain symptom/disorder combinations), we can often simplify by removing variables that are not involved in treatment decisions. Sometimes variables will have to be split or joined to match the expert's intuitions. For example, the original aortic coarctation model had a *Treatment* variable with values *surgery*, *angioplasty*, and *medication*, and a separate variable for *Timing* of the treatment. But the expert had a hard time thinking of these separately, so they were combined, with *Treatment* taking on values such as *surgery in 1 month*. This gives us the model of Figure 16.10.

Assign probabilities. Probabilities can come from patient databases, literature studies, or the expert's subjective assessments. Note that a diagnostic system will reason from symptoms and other observations to the disease or other cause of the problems. Thus, in the early years of building these systems, experts were asked for the probability of a cause given an effect. In general they found this difficult to do, and were better able to assess the probability of an effect given a cause. So modern systems usually assess causal knowledge and encode it directly in the Bayesian network structure of the model, leaving the diagnostic reasoning to the Bayesian network inference algorithms (Shachter and Heckerman, 1987).

Assign utilities. When there are a small number of possible outcomes, they can be enumerated and evaluated individually using the methods of Section 16.3.1. We would create a scale from best to worst outcome and give each a numeric value, for example 0 for death and 1 for complete recovery. We would then place the other outcomes on this scale. This can be done by the expert, but it is better if the patient (or in the case of infants, the patient's parents) can be involved, because different people have different preferences. If there are exponentially many outcomes, we need some way to combine them using multiattribute utility functions. For example, we may say that the costs of various complications are additive.

GOLD STANDARD

Verify and refine the model. To evaluate the system we need a set of correct (input, output) pairs; a so-called **gold standard** to compare against. For medical expert systems this usually means assembling the best available doctors, presenting them with a few cases,

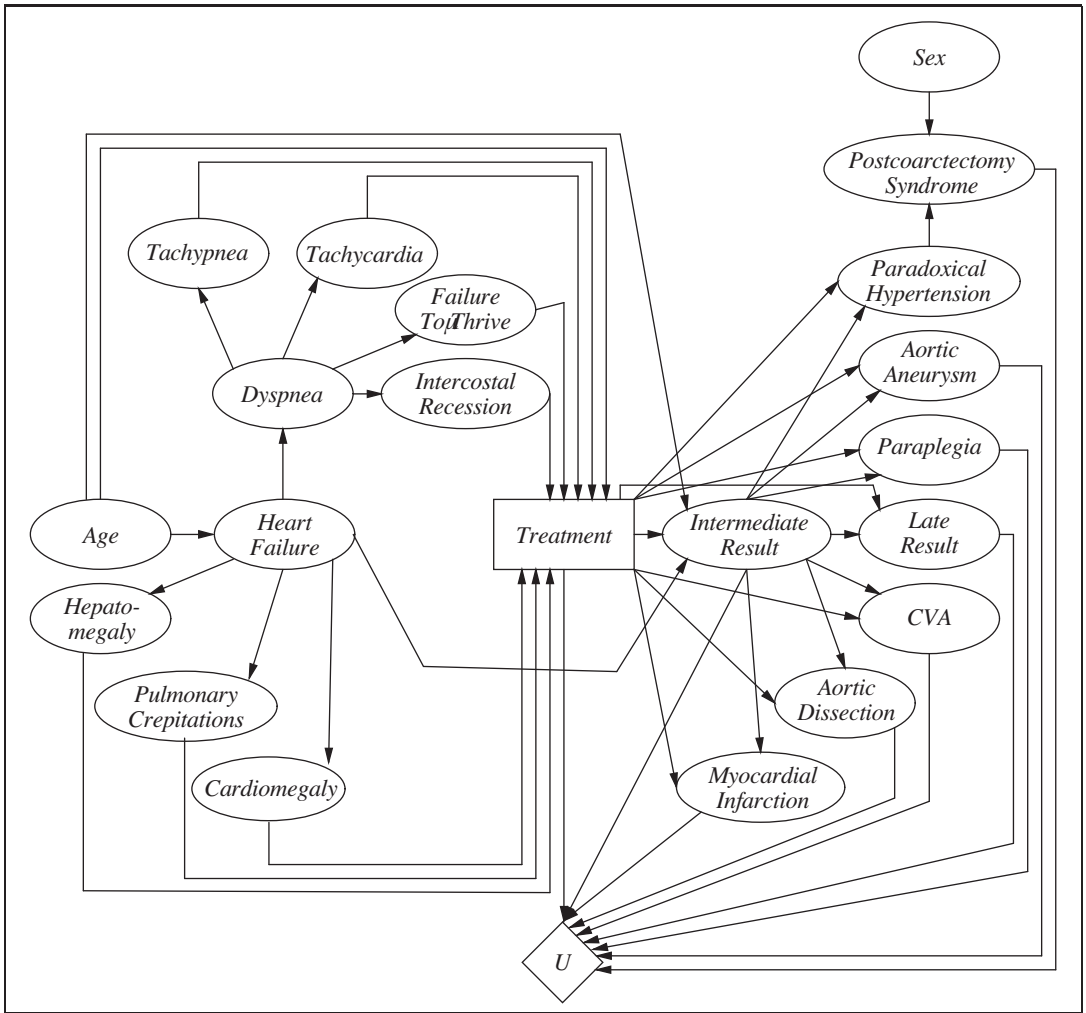


Figure 16.10 Influence diagram for aortic coarctation (courtesy of Peter Lucas).

and asking them for their diagnosis and recommended treatment plan. We then see how well the system matches their recommendations. If it does poorly, we try to isolate the parts that are going wrong and fix them. It can be useful to run the system "backward." Instead of presenting the system with symptoms and asking for a diagnosis, we can present it with a diagnosis such as "heart failure," examine the predicted probability of symptoms such as tachycardia, and compare with the medical literature.

SENSITIVITY ANALYSIS

Perform sensitivity analysis. This important step checks whether the best decision is sensitive to small changes in the assigned probabilities and utilities by systematically varying those parameters and running the evaluation again. If small changes lead to significantly different decisions, then it could be worthwhile to spend more resources to collect better data. If all variations lead to the same decision, then the agent will have more confidence that it is the right decision. Sensitivity analysis is particularly important, because one of the main

criticisms of probabilistic approaches to expert systems is that it is too difficult to assess the numerical probabilities required. Sensitivity analysis often reveals that many of the numbers need be specified only very approximately. For example, we might be uncertain about the conditional probability $P(tachycardia \mid dyspnea)$, but if the optimal decision is reasonably robust to small variations in the probability, then our ignorance is less of a concern.

16.8 SUMMARY

This chapter shows how to combine utility theory with probability to enable an agent to select actions that will maximize its expected performance.

- **Probability theory** describes what an agent should believe on the basis of evidence, **utility theory** describes what an agent wants, and **decision theory** puts the two together to describe what an agent should do.
- We can use decision theory to build a system that makes decisions by considering all possible actions and choosing the one that leads to the best expected outcome. Such a system is known as a **rational agent**.
- Utility theory shows that an agent whose preferences between lotteries are consistent with a set of simple axioms can be described as possessing a utility function; furthermore, the agent selects actions as if maximizing its expected utility.
- **Multiattribute utility theory** deals with utilities that depend on several distinct attributes of states. **Stochastic dominance** is a particularly useful technique for making unambiguous decisions, even without precise utility values for attributes.
- **Decision networks** provide a simple formalism for expressing and solving decision problems. They are a natural extension of Bayesian networks, containing decision and utility nodes in addition to chance nodes.
- Sometimes, solving a problem involves finding more information before making a decision. The **value of information** is defined as the expected improvement in utility compared with making a decision without the information.
- **Expert systems** that incorporate utility information have additional capabilities compared with pure inference systems. In addition to being able to make decisions, they can use the value of information to decide which questions to ask, if any; they can recommend contingency plans; and they can calculate the sensitivity of their decisions to small changes in probability and utility assessments.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

The book *L'art de Penser*, also known as the *Port-Royal Logic* (Arnauld, 1662) states:

> To judge what one must do to obtain a good or avoid an evil, it is necessary to consider not only the good and the evil in itself, but also the probability that it happens or does not happen; and to view geometrically the proportion that all these things have together.

Modern texts talk of *utility* rather than good and evil, but this statement correctly notes that one should multiply utility by probability ("view geometrically") to give expected utility, and maximize that over all outcomes ("all these things") to "judge what one must do." It is remarkable how much this got right, 350 years ago, and only 8 years after Pascal and Fermat showed how to use probability correctly. The Port-Royal Logic also marked the first publication of Pascal's wager.

Daniel Bernoulli (1738), investigating the St. Petersburg paradox (see Exercise 16.3), was the first to realize the importance of preference measurement for lotteries, writing "the *value* of an item must not be based on its *price*, but rather on the *utility* that it yields" (italics his). Utilitarian philosopher Jeremy Bentham (1823) proposed the **hedonic calculus** for weighing "pleasures" and "pains," arguing that all decisions (not just monetary ones) could be reduced to utility comparisons.

The derivation of numerical utilities from preferences was first carried out by Ramsey (1931); the axioms for preference in the present text are closer in form to those rediscovered in *Theory of Games and Economic Behavior* (von Neumann and Morgenstern, 1944). A good presentation of these axioms, in the course of a discussion on risk preference, is given by Howard (1977). Ramsey had derived subjective probabilities (not just utilities) from an agent's preferences; Savage (1954) and Jeffrey (1983) carry out more recent constructions of this kind. Von Winterfeldt and Edwards (1986) provide a modern perspective on decision analysis and its relationship to human preference structures. The micromort utility measure is discussed by Howard (1989). A 1994 survey by the *Economist* set the value of a life at between $750,000 and $2.6 million. However, Richard Thaler (1992) found irrational framing effects on the price one is willing to pay to avoid a risk of death versus the price one is willing to be paid to accept a risk. For a 1/1000 chance, a respondent wouldn't pay more than $200 to remove the risk, but wouldn't accept $50,000 to take on the risk. How much are people willing to pay for a QALY? When it comes down to a specific case of saving oneself or a family member, the number is approximately "whatever I've got." But we can ask at a societal level: suppose there is a vaccine that would yield X QALYs but costs Y dollars; is it worth it? In this case people report a wide range of values from around $10,000 to $150,000 per QALY (Prades *et al.*, 2008). QALYs are much more widely used in medical and social policy decision making than are micromorts; see (Russell, 1990) for a typical example of an argument for a major change in public health policy on grounds of increased expected utility measured in QALYs.

The **optimizer's curse** was brought to the attention of decision analysts in a forceful way by Smith and Winkler (2006), who pointed out that the financial benefits to the client projected by analysts for their proposed course of action almost never materialized. They trace this directly to the bias introduced by selecting an optimal action and show that a more complete Bayesian analysis eliminates the problem. The same underlying concept has been called **post-decision disappointment** by Harrison and March (1984) and was noted in the context of analyzing capital investment projects by Brown (1974). The optimizer's curse is also closely related to the **winner's curse** (Capen *et al.*, 1971; Thaler, 1992), which applies to competitive bidding in auctions: whoever wins the auction is very likely to have overestimated the value of the object in question. Capen *et al.* quote a petroleum engineer on the

POST-DECISION DISAPPOINTMENT

WINNER'S CURSE

topic of bidding for oil-drilling rights: "If one wins a tract against two or three others he may feel fine about his good fortune. But how should he feel if he won against 50 others? Ill."

REGRESSION TO THE MEAN

Finally, behind both curses is the general phenomenon of **regression to the mean**, whereby individuals selected on the basis of exceptional characteristics previously exhibited will, with high probability, become less exceptional in future.

The Allais paradox, due to Nobel Prize-winning economist Maurice Allais (1953) was tested experimentally (Tversky and Kahneman, 1982; Conlisk, 1989) to show that people are consistently inconsistent in their judgments. The Ellsberg paradox on ambiguity aversion was introduced in the Ph.D. thesis of Daniel Ellsberg (Ellsberg, 1962), who went on to become a military analyst at the RAND Corporation and to leak documents known as The Pentagon Papers, which contributed to the end of the Vietnam war and the resignation of President Nixon. Fox and Tversky (1995) describe a further study of ambiguity aversion. Mark Machina (2005) gives an overview of choice under uncertainty and how it can vary from expected utility theory.

There has been a recent outpouring of more-or-less popular books on human irrationality. The best known is *Predictably Irrational* (Ariely, 2009); others include *Sway* (Brafman and Brafman, 2009), *Nudge* (Thaler and Sunstein, 2009), *Kluge* (Marcus, 2009), *How We Decide* (Lehrer, 2009) and *On Being Certain* (Burton, 2009). They complement the classic (Kahneman *et al.*, 1982) and the article that started it all (Kahneman and Tversky, 1979). The field of evolutionary psychology (Buss, 2005), on the other hand, has run counter to this literature, arguing that humans are quite rational in evolutionarily appropriate contexts. Its adherents point out that irrationality is penalized by definition in an evolutionary context and show that in some cases it is an artifact of the experimental setup (Cummins and Allen, 1998). There has been a recent resurgence of interest in Bayesian models of cognition, overturning decades of pessimism (Oaksford and Chater, 1998; Elio, 2002; Chater and Oaksford, 2008).

Keeney and Raiffa (1976) give a thorough introduction to multiattribute utility theory. They describe early computer implementations of methods for eliciting the necessary parameters for a multiattribute utility function and include extensive accounts of real applications of the theory. In AI, the principal reference for MAUT is Wellman's (1985) paper, which includes a system called URP (Utility Reasoning Package) that can use a collection of statements about preference independence and conditional independence to analyze the structure of decision problems. The use of stochastic dominance together with qualitative probability models was investigated extensively by Wellman (1988, 1990a). Wellman and Doyle (1992) provide a preliminary sketch of how a complex set of utility-independence relationships might be used to provide a structured model of a utility function, in much the same way that Bayesian networks provide a structured model of joint probability distributions. Bacchus and Grove (1995, 1996) and La Mura and Shoham (1999) give further results along these lines.

Decision theory has been a standard tool in economics, finance, and management science since the 1950s. Until the 1980s, decision trees were the main tool used for representing simple decision problems. Smith (1988) gives an overview of the methodology of decision analysis. Influence diagrams were introduced by Howard and Matheson (1984), based on earlier work at SRI (Miller *et al.*, 1976). Howard and Matheson's method involved the

derivation of a decision tree from a decision network, but in general the tree is of exponential size. Shachter (1986) developed a method for making decisions based directly on a decision network, without the creation of an intermediate decision tree. This algorithm was also one of the first to provide complete inference for multiply connected Bayesian networks. Zhang *et al.* (1994) showed how to take advantage of conditional independence of information to reduce the size of trees in practice; they use the term *decision network* for networks that use this approach (although others use it as a synonym for influence diagram). Nilsson and Lauritzen (2000) link algorithms for decision networks to ongoing developments in clustering algorithms for Bayesian networks. Koller and Milch (2003) show how influence diagrams can be used to solve games that involve gathering information by opposing players, and Detwarasiti and Shachter (2005) show how influence diagrams can be used as an aid to decision making for a team that shares goals but is unable to share all information perfectly. The collection by Oliver and Smith (1990) has a number of useful articles on decision networks, as does the 1990 special issue of the journal *Networks*. Papers on decision networks and utility modeling also appear regularly in the journals *Management Science* and *Decision Analysis*.

The theory of information value was explored first in the context of statistical experiments, where a quasi-utility (entropy reduction) was used (Lindley, 1956). The Russian control theorist Ruslan Stratonovich (1965) developed the more general theory presented here, in which information has value by virtue of its ability to affect decisions. Stratonovich's work was not known in the West, where Ron Howard (1966) pioneered the same idea. His paper ends with the remark "If information value theory and associated decision theoretic structures do not in the future occupy a large part of the education of engineers, then the engineering profession will find that its traditional role of managing scientific and economic resources for the benefit of man has been forfeited to another profession." To date, the implied revolution in managerial methods has not occurred.

Recent work by Krause and Guestrin (2009) shows that computing the exact nonmyopic value of information is intractable even in polytree networks. There are other cases—more restricted than general value of information—in which the myopic algorithm does provide a provably good approximation to the optimal sequence of observations (Krause *et al.*, 2008). In some cases—for example, looking for treasure buried in one of n places—ranking experiments in order of success probability divided by cost gives an optimal solution (Kadane and Simon, 1977).

Surprisingly few early AI researchers adopted decision-theoretic tools after the early applications in medical decision making described in Chapter 13. One of the few exceptions was Jerry Feldman, who applied decision theory to problems in vision (Feldman and Yakimovsky, 1974) and planning (Feldman and Sproull, 1977). After the resurgence of interest in probabilistic methods in AI in the 1980s, decision-theoretic expert systems gained widespread acceptance (Horvitz *et al.*, 1988; Cowell *et al.*, 2002). In fact, from 1991 onward, the cover design of the journal *Artificial Intelligence* has depicted a decision network, although some artistic license appears to have been taken with the direction of the arrows.

EXERCISES

16.1 (Adapted from David Heckerman.) This exercise concerns the **Almanac Game**, which is used by decision analysts to calibrate numeric estimation. For each of the questions that follow, give your best guess of the answer, that is, a number that you think is as likely to be too high as it is to be too low. Also give your guess at a 25th percentile estimate, that is, a number that you think has a 25% chance of being too high, and a 75% chance of being too low. Do the same for the 75th percentile. (Thus, you should give three estimates in all—low, median, and high—for each question.)

a. Number of passengers who flew between New York and Los Angeles in 1989.

b. Population of Warsaw in 1992.

c. Year in which Coronado discovered the Mississippi River.

d. Number of votes received by Jimmy Carter in the 1976 presidential election.

e. Age of the oldest living tree, as of 2002.

f. Height of the Hoover Dam in feet.

g. Number of eggs produced in Oregon in 1985.

h. Number of Buddhists in the world in 1992.

i. Number of deaths due to AIDS in the United States in 1981.

j. Number of U.S. patents granted in 1901.

The correct answers appear after the last exercise of this chapter. From the point of view of decision analysis, the interesting thing is not how close your median guesses came to the real answers, but rather how often the real answer came within your 25% and 75% bounds. If it was about half the time, then your bounds are accurate. But if you're like most people, you will be more sure of yourself than you should be, and fewer than half the answers will fall within the bounds. With practice, you can calibrate yourself to give realistic bounds, and thus be more useful in supplying information for decision making. Try this second set of questions and see if there is any improvement:

a. Year of birth of Zsa Zsa Gabor.

b. Maximum distance from Mars to the sun in miles.

c. Value in dollars of exports of wheat from the United States in 1992.

d. Tons handled by the port of Honolulu in 1991.

e. Annual salary in dollars of the governor of California in 1993.

f. Population of San Diego in 1990.

g. Year in which Roger Williams founded Providence, Rhode Island.

h. Height of Mt. Kilimanjaro in feet.

i. Length of the Brooklyn Bridge in feet.

j. Number of deaths due to automobile accidents in the United States in 1992.

16.2 Chris considers four used cars before buying the one with maximum expected utility. Pat considers ten cars and does the same. All other things being equal, which one is more likely to have the better car? Which is more likely to be disappointed with their car's quality? By how much (in terms of standard deviations of expected quality)?

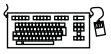
16.3 In 1713, Nicolas Bernoulli stated a puzzle, now called the St. Petersburg paradox, which works as follows. You have the opportunity to play a game in which a fair coin is tossed repeatedly until it comes up heads. If the first heads appears on the nth toss, you win 2^n dollars.

a. Show that the expected monetary value of this game is infinite.

b. How much would you, personally, pay to play the game?

c. Nicolas's cousin Daniel Bernoulli resolved the apparent paradox in 1738 by suggesting that the utility of money is measured on a logarithmic scale (i.e., $U(S_n) = a \log_2 n + b$, where S_n is the state of having $\$n$). What is the expected utility of the game under this assumption?

d. What is the maximum amount that it would be rational to pay to play the game, assuming that one's initial wealth is $\$k$?



16.4 Write a computer program to automate the process in Exercise 16.9. Try your program out on several people of different net worth and political outlook. Comment on the consistency of your results, both for an individual and across individuals.

16.5 The Surprise Candy Company makes candy in two flavors: 70% are strawberry flavor and 30% are anchovy flavor. Each new piece of candy starts out with a round shape; as it moves along the production line, a machine randomly selects a certain percentage to be trimmed into a square; then, each piece is wrapped in a wrapper whose color is chosen randomly to be red or brown. 80% of the strawberry candies are round and 80% have a red wrapper, while 90% of the anchovy candies are square and 90% have a brown wrapper. All candies are sold individually in sealed, identical, black boxes.

Now you, the customer, have just bought a Surprise candy at the store but have not yet opened the box. Consider the three Bayes nets in Figure 16.11.

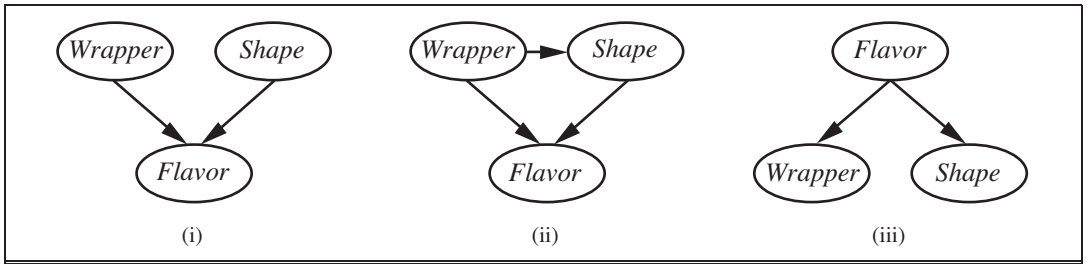


Figure 16.11 Three proposed Bayes nets for the Surprise Candy problem, Exercise 16.5.

a. Which network(s) can correctly represent $\mathbf{P}(Flavor, Wrapper, Shape)$?

b. Which network is the best representation for this problem?

c. Does network (i) assert that $\mathbf{P}(Wrapper|Shape) = \mathbf{P}(Wrapper)$?

d. What is the probability that your candy has a red wrapper?

e. In the box is a round candy with a red wrapper. What is the probability that its flavor is strawberry?

f. A unwrapped strawberry candy is worth s on the open market and an unwrapped anchovy candy is worth a. Write an expression for the value of an unopened candy box.

g. A new law prohibits trading of unwrapped candies, but it is still legal to trade wrapped candies (out of the box). Is an unopened candy box now worth more than less than, or the same as before?

16.6 Prove that the judgments $B \succ A$ and $C \succ D$ in the Allais paradox (page 620) violate the axiom of substitutability.

16.7 Consider the Allais paradox described on page 620: an agent who prefers B over A (taking the sure thing), and C over D (taking the higher EMV) is not acting rationally, according to utility theory. Do you think this indicates a problem for the agent, a problem for the theory, or no problem at all? Explain.

16.8 Tickets to a lottery cost \$1. There are two possible prizes: a \$10 payoff with probability 1/50, and a \$1,000,000 payoff with probability 1/2,000,000. What is the expected monetary value of a lottery ticket? When (if ever) is it rational to buy a ticket? Be precise—show an equation involving utilities. You may assume current wealth of $\$k$ and that $U(S_k) = 0$. You may also assume that $U(S_{k+10}) = 10 \times U(S_{k+1})$, but you may not make any assumptions about $U(S_{k+1{,}000{,}000})$. Sociological studies show that people with lower income buy a disproportionate number of lottery tickets. Do you think this is because they are worse decision makers or because they have a different utility function? Consider the value of contemplating the possibility of winning the lottery versus the value of contemplating becoming an action hero while watching an adventure movie.

16.9 Assess your own utility for different incremental amounts of money by running a series of preference tests between some definite amount M_1 and a lottery $[p, M_2; (1-p), 0]$. Choose different values of M_1 and M_2, and vary p until you are indifferent between the two choices. Plot the resulting utility function.

16.10 How much is a micromort worth to you? Devise a protocol to determine this. Ask questions based both on paying to avoid risk and being paid to accept risk.

16.11 Let continuous variables $X_1, \ldots, X_k$ be independently distributed according to the same probability density function $f(x)$. Prove that the density function for $\max\{X_1, \ldots, X_k\}$ is given by $kf(x)(F(x))^{k-1}$, where F is the cumulative distribution for f.

16.12 Economists often make use of an exponential utility function for money: $U(x) = -e^{x/R}$, where R is a positive constant representing an individual's risk tolerance. Risk tolerance reflects how likely an individual is to accept a lottery with a particular expected monetary value (EMV) versus some certain payoff. As R (which is measured in the same units as x) becomes larger, the individual becomes less risk-averse.

a. Assume Mary has an exponential utility function with $R = \$500$. Mary is given the choice between receiving \$500 with certainty (probability 1) or participating in a lottery which has a 60% probability of winning \$5000 and a 40% probability of winning nothing. Assuming Marry acts rationally, which option would she choose? Show how you derived your answer.

b. Consider the choice between receiving \$100 with certainty (probability 1) or participating in a lottery which has a 50% probability of winning \$500 and a 50% probability of winning nothing. Approximate the value of R (to 3 significant digits) in an exponential utility function that would cause an individual to be indifferent to these two alternatives. (You might find it helpful to write a short program to help you solve this problem.)

16.13 Repeat Exercise 16.16, using the action-utility representation shown in Figure 16.7.

16.14 For either of the airport-siting diagrams from Exercises 16.16 and 16.13, to which conditional probability table entry is the utility most sensitive, given the available evidence?

16.15 Consider a student who has the choice to buy or not buy a textbook for a course. We'll model this as a decision problem with one Boolean decision node, B, indicating whether the agent chooses to buy the book, and two Boolean chance nodes, M, indicating whether the student has mastered the material in the book, and P, indicating whether the student passes the course. Of course, there is also a utility node, U. A certain student, Sam, has an additive utility function: 0 for not buying the book and -\$100 for buying it; and \$2000 for passing the course and 0 for not passing. Sam's conditional probability estimates are as follows:

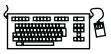
$$P(p|b,m) = 0.9 \qquad P(m|b) = 0.9$$
$$P(p|b,\neg m) = 0.5 \qquad P(m|\neg b) = 0.7$$
$$P(p|\neg b,m) = 0.8$$
$$P(p|\neg b,\neg m) = 0.3$$

You might think that P would be independent of B given M, But this course has an open-book final—so having the book helps.

a. Draw the decision network for this problem.

b. Compute the expected utility of buying the book and of not buying it.

c. What should Sam do?



16.16 This exercise completes the analysis of the airport-siting problem in Figure 16.6.

a. Provide reasonable variable domains, probabilities, and utilities for the network, assuming that there are three possible sites.

b. Solve the decision problem.

c. What happens if changes in technology mean that each aircraft generates half the noise?

d. What if noise avoidance becomes three times more important?

e. Calculate the VPI for $AirTraffic$, $Litigation$, and $Construction$ in your model.

16.17 (Adapted from Pearl (1988).) A used-car buyer can decide to carry out various tests with various costs (e.g., kick the tires, take the car to a qualified mechanic) and then, depending on the outcome of the tests, decide which car to buy. We will assume that the buyer is deciding whether to buy car c_1, that there is time to carry out at most one test, and that t_1 is the test of c_1 and costs \$50.

A car can be in good shape (quality q^+) or bad shape (quality q^-), and the tests might help indicate what shape the car is in. Car c_1 costs \$1,500, and its market value is \$2,000 if it is in good shape; if not, \$700 in repairs will be needed to make it in good shape. The buyer's estimate is that c_1 has a 70% chance of being in good shape.

a. Draw the decision network that represents this problem.

b. Calculate the expected net gain from buying c_1, given no test.

c. Tests can be described by the probability that the car will pass or fail the test given that the car is in good or bad shape. We have the following information:
$P(pass(c_1, t_1)|q^+(c_1)) = 0.8$
$P(pass(c_1, t_1)|q^-(c_1)) = 0.35$
Use Bayes' theorem to calculate the probability that the car will pass (or fail) its test and hence the probability that it is in good (or bad) shape given each possible test outcome.

d. Calculate the optimal decisions given either a pass or a fail, and their expected utilities.

e. Calculate the value of information of the test, and derive an optimal conditional plan for the buyer.

16.18 Recall the definition of *value of information* in Section 16.6.

a. Prove that the value of information is nonnegative and order independent.

b. Explain why it is that some people would prefer not to get some information—for example, not wanting to know the sex of their baby when an ultrasound is done.

SUBMODULARITY

c. A function f on sets is **submodular** if, for any element x and any sets A and B such that $A \subseteq B$, adding x to A gives a greater increase in f than adding x to B:

$$A \subseteq B \Rightarrow (f(A \cup \{x\}) - f(A)) \geq (f(B \cup \{x\}) - f(B)) .$$

Submodularity captures the intuitive notion of *diminishing returns*. Is the value of information, viewed as a function f on sets of possible observations, submodular? Prove this or find a counterexample.

The answers to Exercise 16.1 (where M stands for million): First set: 3M, 1.6M, 1541, 41M, 4768, 221, 649M, 295M, 132, 25,546. Second set: 1917, 155M, 4,500M, 11M, 120,000, 1.1M, 1636, 19,340, 1,595, 41,710.

17 MAKING COMPLEX DECISIONS

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

SEQUENTIAL DECISION PROBLEM

In this chapter, we address the computational issues involved in making decisions in a stochastic environment. 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 are concerned here with **sequential decision problems**, in which the agent's utility depends on a sequence of decisions. Sequential decision problems incorporate utilities, uncertainty, and sensing, and include search and planning problems as special cases. 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 17.4.3 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.

The second part of the chapter covers environments with multiple agents. In such environments, the notion of optimal behavior is complicated by the interactions among the agents. Section 17.5 introduces the main ideas of **game theory**, including the idea that rational agents might need to behave randomly. Section 17.6 looks at how multiagent systems can be designed so that multiple agents can achieve a common goal.

17.1 SEQUENTIAL DECISION PROBLEMS

Suppose that an agent is situated in the 4×3 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. Just as for search problems, the actions available to the agent in each state are given by $\text{ACTIONS}(s)$, sometimes abbreviated to $A(s)$; in the 4×3 environment, the actions in every state are *Up*, *Down*, *Left*, and *Right*. We assume for now that the environment is **fully observable**, so that the agent always knows where it is.

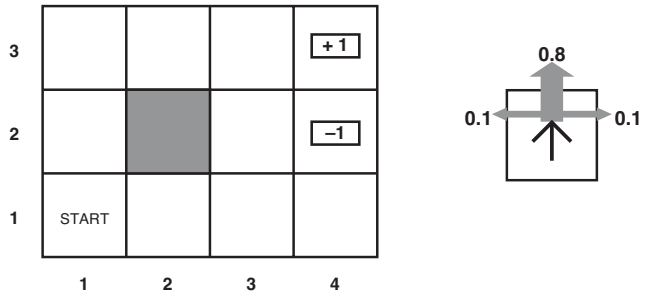


Figure 17.1 (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision 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.)

As in Chapter 3, the **transition model** (or just "model," whenever no confusion can arise) describes the outcome of each action in each state. Here, the outcome is stochastic, so we write $P(s' \mid s, a)$ 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 $P(s' \mid s, a)$ as a big three-dimensional table containing probabilities. Later, in Section 17.4.3, we will see that the transition model can be represented as a **dynamic Bayesian network**, just as in Chapter 15.

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 investigate how such utility functions can be specified in general; for now,
REWARD we 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 leave as soon as possible.

MARKOV DECISION PROCESS

To sum up: a sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a **Markov decision process**, or **MDP**, and consists of a set of states (with an initial state s_0); a set $\text{ACTIONS}(s)$ of actions in each state; a transition model $P(s' \mid s, a)$; and a reward function $R(s)$.[1]

POLICY

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**. It is traditional to denote a policy by π, and $\pi(s)$ is the action recommended by the policy π 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.

OPTIMAL POLICY

Each time a given policy is executed starting from the initial state, the stochastic nature of the environment may 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 π^* to denote an optimal policy. Given π^*, 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.

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 shortcut and thereby risking entering (4,2).

The balance of risk and reward changes depending 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 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),

[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, as shown in Exercise 17.4.

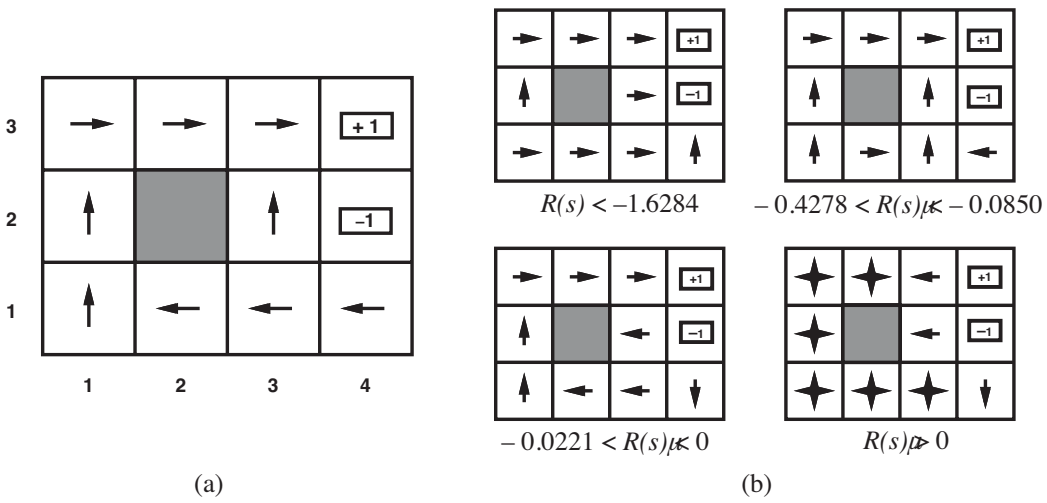


Figure 17.2 (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$.

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 17.5 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 describe two of the most important algorithm families. First, however, we must complete our investigation of utilities and policies for sequential decision problems.

17.1.1 Utilities over time

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 for the utility function on environment histories, which we write as $U_h([s_0, s_1, \ldots, s_n])$. Our analysis draws on **multiattribute utility theory** (Section 16.4) and is somewhat technical; the impatient reader may wish to skip to the next section.

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 4×3 world of Figure 17.1, 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,*

NONSTATIONARY POLICY

STATIONARY POLICY

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 deal mainly with the infinite-horizon case in this chapter. (We will see later that for partially observable environments, the infinite-horizon case is not so simple.) 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.

The next question we must decide is how to calculate the utility of state sequences. In the terminology of multiattribute utility theory, each state s_i can be 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 instead. Stationarity is a fairly innocuous-looking assumption with very strong consequences: it turns out that under stationarity there are just two coherent ways to assign utilities to sequences:

STATIONARY PREFERENCE

ADDITIVE REWARD

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 4×3 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 3).

DISCOUNTED REWARD

2. **Discounted rewards**: The utility of a state sequence is

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots ,$$

DISCOUNT FACTOR

where the **discount factor** γ is a number between 0 and 1. The discount factor describes the preference of an agent for current rewards over future rewards. When γ is close to 0, rewards in the distant future are viewed as insignificant. When γ 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 γ is equivalent to an interest rate of $(1/\gamma) - 1$.

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

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, undiscounted rewards will generally be

infinite. While we can agree that $+\infty$ is better than $-\infty$, comparing two state sequences with $+\infty$ 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 $\gamma < 1$ and rewards are bounded by $\pm R_{\max}$, 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

3. Infinite sequences can be compared in terms of the **average reward** obtained per time step. Suppose that square (1,1) in the 4×3 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.

AVERAGE REWARD

In sum, discounted rewards present the fewest difficulties in evaluating state sequences.

17.1.2 Optimal policies and the utilities of states

Having decided that the utility of a given state sequence is the sum of discounted rewards obtained during the sequence, we can compare policies by comparing the *expected* utilities obtained when executing them. We assume the agent is in some initial state s and define S_t (a random variable) to be the state the agent reaches at time t when executing a particular policy π. (Obviously, $S_0 = s$, the state the agent is in now.) The probability distribution over state sequences $S_1, S_2, \ldots$, is determined by the initial state s, the policy π, and the transition model for the environment.

The expected utility obtained by executing π starting in s is given by
$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right] \ , \qquad (17.2)$$
where the expectation is with respect to the probability distribution over state sequences determined by s and π. Now, out of all the policies the agent could choose to execute starting in s, one (or more) will have higher expected utilities than all the others. We'll use π_s^* to denote one of these policies:
$$\pi_s^* = \operatorname*{argmax}_{\pi} U^{\pi}(s) \ . \qquad (17.3)$$

Remember that π_s^* is a policy, so it recommends an action for every state; its connection with s in particular is that it's an optimal policy when s is the starting state. A remarkable consequence of using discounted utilities with infinite horizons is that the optimal policy is *independent* of the starting state. (Of course, the *action sequence* won't be independent; remember that a policy is a function specifying an action for each state.) This fact seems intuitively obvious: if policy π_a^* is optimal starting in a and policy π_b^* is optimal starting in b, then, when they reach a third state c, there's no good reason for them to disagree with each other, or with π_c^*, about what to do next.[2] So we can simply write π^* for an optimal policy.

Given this definition, the true utility of a state is just $U^{\pi^*}(s)$—that is, the expected sum of discounted rewards if the agent executes an optimal policy. We write this as $U(s)$, matching the notation used in Chapter 16 for the utility of an outcome. Notice that $U(s)$ and $R(s)$ are quite different quantities; $R(s)$ is the "short term" reward for being in s, whereas $U(s)$ is the "long term" total reward from s onward. Figure 17.3 shows the utilities for the 4×3 world. Notice that the utilities are higher for states closer to the +1 exit, because fewer steps are required to reach the exit.

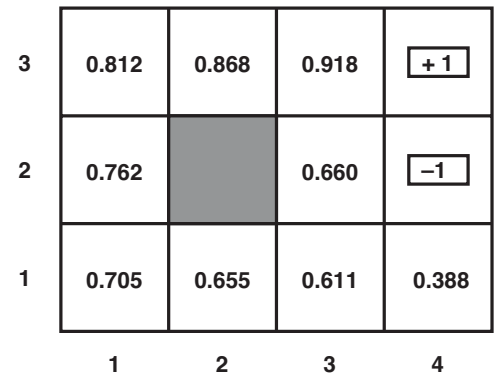


Figure 17.3 The utilities of the states in the 4×3 world, calculated with $\gamma = 1$ and $R(s) = -0.04$ for nonterminal states.

The utility function $U(s)$ allows the agent to select actions by using the principle of maximum expected utility from Chapter 16—that is, choose the action that maximizes the expected utility of the subsequent state:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \ . \tag{17.4}$$

The next two sections describe algorithms for finding optimal policies.

[2] Although this seems obvious, it does not hold for finite-horizon policies or for other ways of combining rewards over time. The proof follows directly from the uniqueness of the utility function on states, as shown in Section 17.2.

17.2 VALUE ITERATION

VALUE ITERATION

In this section, we present an algorithm, called **value iteration**, for calculating an optimal policy. The basic idea is to calculate the utility of each state and then use the state utilities to select an optimal action in each state.

17.2.1 The Bellman equation for utilities

Section 17.1.2 defined the utility of being in a state as the expected sum of discounted rewards from that point onwards. From this, it follows that there is a direct relationship between the utility of a state and the utility of its neighbors: *the utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.* That is, the utility of a state is given by

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s') \ . \tag{17.5}$$

BELLMAN EQUATION

This is called the **Bellman equation**, after Richard Bellman (1957). The utilities of the states—defined by Equation (17.2) as the expected utility of subsequent state sequences—are solutions of the set of Bellman equations. In fact, they are the *unique* solutions, as we show in Section 17.2.3.

Let us look at one of the Bellman equations for the 4×3 world. The equation for the state (1,1) is

$$\begin{aligned} U(1,1) = -0.04 + \gamma \ \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (\mathit{Up}) \\ & 0.9U(1,1) + 0.1U(1,2), && (\mathit{Left}) \\ & 0.9U(1,1) + 0.1U(2,1), && (\mathit{Down}) \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1) \ ]. && (\mathit{Right}) \end{aligned}$$

When we plug in the numbers from Figure 17.3, we find that *Up* is the best action.

17.2.2 The value iteration algorithm

The Bellman equation is the basis of the value iteration algorithm for solving MDPs. If there are n possible states, then there are n Bellman equations, one for each state. The n equations contain n unknowns—the utilities of the states. So we would like to solve these simultaneous equations to find the utilities. There is one problem: the equations are *nonlinear*, because the "max" operator is not a linear operator. Whereas systems of linear equations can be solved quickly using linear algebra techniques, systems of nonlinear equations are more problematic. One thing to try is an *iterative* approach. We start with arbitrary initial values for the utilities, calculate the right-hand side of the equation, and plug it into the left-hand side—thereby updating the utility of each state from the utilities of its neighbors. We repeat this until we reach an equilibrium. Let $U_i(s)$ be the utility value for state s at the ith iteration. The iteration step, called a **Bellman update**, looks like this:

BELLMAN UPDATE

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U_i(s') \ , \tag{17.6}$$

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum change in the utility of any state in an iteration

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return U

Figure 17.4 The value iteration algorithm for calculating utilities of states. The termination condition is from Equation (17.8).

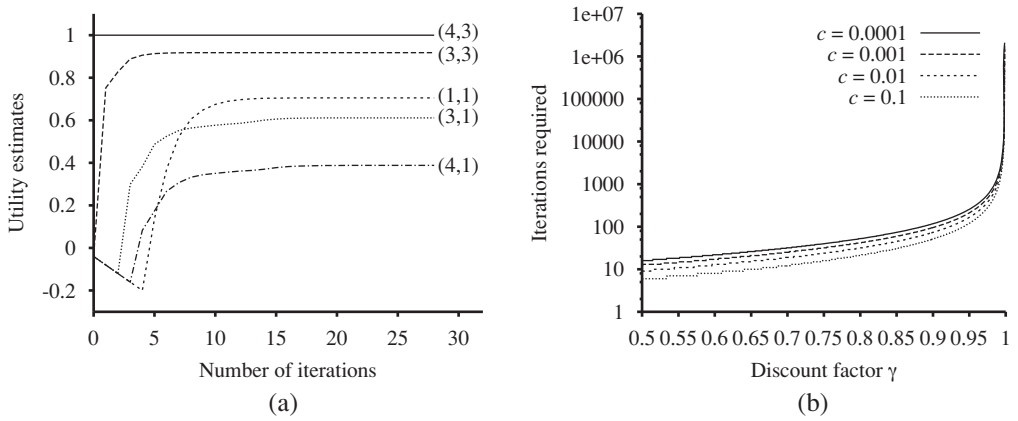


Figure 17.5 (a) Graph showing the evolution of the utilities of selected states using value iteration. (b) The number of value iterations k required to guarantee an error of at most $\epsilon = c \cdot R_{\max}$, for different values of c, as a function of the discount factor γ.

where the update is assumed to be applied simultaneously to all the states at each iteration. If we apply the Bellman update infinitely often, we are guaranteed to reach an equilibrium (see Section 17.2.3), in which case the final utility values must be solutions to the Bellman equations. In fact, they are also the *unique* solutions, and the corresponding policy (obtained using Equation (17.4)) is optimal. The algorithm, called VALUE-ITERATION, is shown in Figure 17.4.

We can apply value iteration to the 4×3 world in Figure 17.1(a). Starting with initial values of zero, the utilities evolve as shown in Figure 17.5(a). Notice how the states at differ-

ent distances from (4,3) accumulate negative reward until a path is found to (4,3), whereupon the utilities start to increase. We can think of the value iteration algorithm as *propagating information* through the state space by means of local updates.

17.2.3 Convergence of value iteration

We said that value iteration eventually converges to a unique set of solutions of the Bellman equations. In this section, we explain why this happens. We introduce some useful mathematical ideas along the way, and we obtain some methods for assessing the error in the utility function returned when the algorithm is terminated early; this is useful because it means that we don't have to run forever. This section is quite technical.

CONTRACTION

The basic concept used in showing that value iteration converges is the notion of a **contraction**. Roughly speaking, a contraction is a function of one argument that, when applied to two different inputs in turn, produces two output values that are "closer together," by at least some constant factor, than the original inputs. For example, the function "divide by two" is a contraction, because, after we divide any two numbers by two, their difference is halved. Notice that the "divide by two" function has a fixed point, namely zero, that is unchanged by the application of the function. From this example, we can discern two important properties of contractions:

- A contraction has only one fixed point; if there were two fixed points they would not get closer together when the function was applied, so it would not be a contraction.
- When the function is applied to any argument, the value must get closer to the fixed point (because the fixed point does not move), so repeated application of a contraction always reaches the fixed point in the limit.

Now, suppose we view the Bellman update (Equation (17.6)) as an operator B that is applied simultaneously to update the utility of every state. Let U_i denote the vector of utilities for all the states at the ith iteration. Then the Bellman update equation can be written as

$$U_{i+1} \leftarrow B\,U_i\ .$$

MAX NORM

Next, we need a way to measure distances between utility vectors. We will use the **max norm**, which measures the "length" of a vector by the absolute value of its biggest component:

$$||U|| = \max_s |U(s)|\ .$$

With this definition, the "distance" between two vectors, $||U - U'||$, is the maximum difference between any two corresponding elements. The main result of this section is the following: *Let U_i and U_i' be any two utility vectors. Then we have*

$$||B\,U_i - B\,U_i'|| \le \gamma\,||U_i - U_i'||\ . \tag{17.7}$$

That is, the Bellman update is a contraction by a factor of γ on the space of utility vectors. (Exercise 17.6 provides some guidance on proving this claim.) Hence, from the properties of contractions in general, it follows that value iteration always converges to a unique solution of the Bellman equations whenever $\gamma < 1$.

We can also use the contraction property to analyze the *rate* of convergence to a solution. In particular, we can replace U_i' in Equation (17.7) with the *true* utilities U, for which $B\,U = U$. Then we obtain the inequality

$$||B\,U_i - U|| \leq \gamma\,||U_i - U|| \;.$$

So, if we view $||U_i - U||$ as the *error* in the estimate U_i, we see that the error is reduced by a factor of at least γ on each iteration. This means that value iteration converges exponentially fast. We can calculate the number of iterations required to reach a specified error bound ϵ as follows: First, recall from Equation (17.1) that the utilities of all states are bounded by $\pm R_{\max}/(1-\gamma)$. This means that the maximum initial error $||U_0 - U|| \leq 2R_{\max}/(1-\gamma)$. Suppose we run for N iterations to reach an error of at most ϵ. Then, because the error is reduced by at least γ each time, we require $\gamma^N \cdot 2R_{\max}/(1-\gamma) \leq \epsilon$. Taking logs, we find

$$N = \lceil \log(2R_{\max}/\epsilon(1-\gamma))/\log(1/\gamma) \rceil$$

iterations suffice. Figure 17.5(b) shows how N varies with γ, for different values of the ratio $\epsilon/R_{\max}$. The good news is that, because of the exponentially fast convergence, N does not depend much on the ratio $\epsilon/R_{\max}$. The bad news is that N grows rapidly as γ becomes close to 1. We can get fast convergence if we make γ small, but this effectively gives the agent a short horizon and could miss the long-term effects of the agent's actions.

The error bound in the preceding paragraph gives some idea of the factors influencing the run time of the algorithm, but is sometimes overly conservative as a method of deciding when to stop the iteration. For the latter purpose, we can use a bound relating the error to the size of the Bellman update on any given iteration. From the contraction property (Equation (17.7)), it can be shown that if the update is small (i.e., no state's utility changes by much), then the error, compared with the true utility function, also is small. More precisely,

$$\text{if} \quad ||U_{i+1} - U_i|| < \epsilon(1-\gamma)/\gamma \quad \text{then} \quad ||U_{i+1} - U|| < \epsilon \;. \tag{17.8}$$

This is the termination condition used in the VALUE-ITERATION algorithm of Figure 17.4.

So far, we have analyzed the error in the utility function returned by the value iteration algorithm. *What the agent really cares about, however, is how well it will do if it makes its decisions on the basis of this utility function.* Suppose that after i iterations of value iteration, the agent has an estimate U_i of the true utility U and obtains the MEU policy π_i based on one-step look-ahead using U_i (as in Equation (17.4)). Will the resulting behavior be nearly as good as the optimal behavior? This is a crucial question for any real agent, and it turns out that the answer is yes. $U^{\pi_i}(s)$ is the utility obtained if π_i is executed starting in s, and the

POLICY LOSS

policy loss $||U^{\pi_i} - U||$ is the most the agent can lose by executing π_i instead of the optimal policy π^*. The policy loss of π_i is connected to the error in U_i by the following inequality:

$$\text{if} \quad ||U_i - U|| < \epsilon \quad \text{then} \quad ||U^{\pi_i} - U|| < 2\epsilon\gamma/(1-\gamma) \;. \tag{17.9}$$

In practice, it often occurs that π_i becomes optimal long before U_i has converged. Figure 17.6 shows how the maximum error in U_i and the policy loss approach zero as the value iteration process proceeds for the 4×3 environment with $\gamma = 0.9$. The policy π_i is optimal when $i = 4$, even though the maximum error in U_i is still 0.46.

Now we have everything we need to use value iteration in practice. We know that it converges to the correct utilities, we can bound the error in the utility estimates if we

stop after a finite number of iterations, and we can bound the policy loss that results from executing the corresponding MEU policy. As a final note, all of the results in this section depend on discounting with $\gamma < 1$. If $\gamma = 1$ and the environment contains terminal states, then a similar set of convergence results and error bounds can be derived whenever certain technical conditions are satisfied.

17.3 POLICY ITERATION

In the previous section, we observed that it is possible to get an optimal policy even when the utility function estimate is inaccurate. If one action is clearly better than all others, then the exact magnitude of the utilities on the states involved need not be precise. This insight suggests an alternative way to find optimal policies. The **policy iteration** algorithm alternates the following two steps, beginning from some initial policy π_0:

POLICY ITERATION

POLICY EVALUATION

POLICY IMPROVEMENT

- **Policy evaluation**: given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed.
- **Policy improvement**: Calculate a new MEU policy π_{i+1}, using one-step look-ahead based on U_i (as in Equation (17.4)).

The algorithm terminates when the policy improvement step yields no change in the utilities. At this point, we know that the utility function U_i is a fixed point of the Bellman update, so it is a solution to the Bellman equations, and π_i must be an optimal policy. Because there are only finitely many policies for a finite state space, and each iteration can be shown to yield a better policy, policy iteration must terminate. The algorithm is shown in Figure 17.7.

The policy improvement step is obviously straightforward, but how do we implement the POLICY-EVALUATION routine? It turns out that doing so is much simpler than solving the standard Bellman equations (which is what value iteration does), because the action in each state is fixed by the policy. At the ith iteration, the policy π_i specifies the action $\pi_i(s)$ in

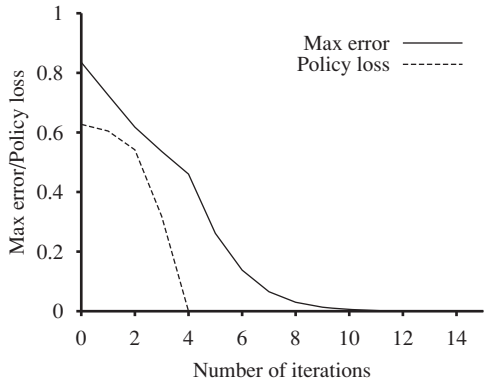


Figure 17.6 The maximum error $||U_i - U||$ of the utility estimates and the policy loss $||U^{\pi_i} - U||$, as a function of the number of iterations of value iteration.

state s. This means that we have a simplified version of the Bellman equation (17.5) relating the utility of s (under π_i) to the utilities of its neighbors:

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s') \ . \tag{17.10}$$

For example, suppose π_i is the policy shown in Figure 17.2(a). Then we have $\pi_i(1,1) = Up$, $\pi_i(1,2) = Up$, and so on, and the simplified Bellman equations are

$$\begin{aligned} U_i(1,1) &= -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1) \ , \\ U_i(1,2) &= -0.04 + 0.8U_i(1,3) + 0.2U_i(1,2) \ , \\ &\vdots \end{aligned}$$

The important point is that these equations are *linear*, because the "max" operator has been removed. For n states, we have n linear equations with n unknowns, which can be solved exactly in time $O(n^3)$ by standard linear algebra methods.

For small state spaces, policy evaluation using exact solution methods is often the most efficient approach. For large state spaces, $O(n^3)$ time might be prohibitive. Fortunately, it is not necessary to do *exact* policy evaluation. Instead, we can perform some number of simplified value iteration steps (simplified because the policy is fixed) to give a reasonably good approximation of the utilities. The simplified Bellman update for this process is

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s') \ ,$$

MODIFIED POLICY ITERATION

and this is repeated k times to produce the next utility estimate. The resulting algorithm is called **modified policy iteration**. It is often much more efficient than standard policy iteration or value iteration.

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] \ > \ \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

The algorithms we have described so far require updating the utility or policy for all states at once. It turns out that this is not strictly necessary. In fact, on each iteration, we can pick *any subset* of states and apply *either* kind of updating (policy improvement or simplified value iteration) to that subset. This very general algorithm is called **asynchronous policy iteration**. Given certain conditions on the initial policy and initial utility function, asynchronous policy iteration is guaranteed to converge to an optimal policy. The freedom to choose any states to work on means that we can design much more efficient heuristic algorithms—for example, algorithms that concentrate on updating the values of states that are likely to be reached by a good policy. This makes a lot of sense in real life: if one has no intention of throwing oneself off a cliff, one should not spend time worrying about the exact value of the resulting states.

ASYNCHRONOUS POLICY ITERATION

17.4 PARTIALLY OBSERVABLE MDPS

The description of Markov decision processes in Section 17.1 assumed that the environment was **fully observable**. With this assumption, the agent always knows which state it is in. This, combined with the Markov assumption for the transition model, means that the optimal policy depends only on the current state. When the environment is only **partially observable**, the situation is, one might say, much less clear. The agent does not necessarily know which state it is in, so it cannot execute the action $\pi(s)$ recommended for that state. Furthermore, the utility of a state s and the optimal action in s depend not just on s, but also on *how much the agent knows* when it is in s. For these reasons, **partially observable MDPs** (or POMDPs—pronounced "pom-dee-pees") are usually viewed as much more difficult than ordinary MDPs. We cannot avoid POMDPs, however, because the real world is one.

PARTIALLY OBSERVABLE MDP

17.4.1 Definition of POMDPs

To get a handle on POMDPs, we must first define them properly. A POMDP has the same elements as an MDP—the transition model $P(s' \mid s, a)$, actions $A(s)$, and reward function $R(s)$—but, like the partially observable search problems of Section 4.4, it also has a **sensor model** $P(e \mid s)$. Here, as in Chapter 15, the sensor model specifies the probability of perceiving evidence e in state s.[3] For example, we can convert the 4×3 world of Figure 17.1 into a POMDP by adding a noisy or partial sensor instead of assuming that the agent knows its location exactly. Such a sensor might measure the *number of adjacent walls*, which happens to be 2 in all the nonterminal squares except for those in the third column, where the value is 1; a noisy version might give the wrong value with probability 0.1.

In Chapters 4 and 11, we studied nondeterministic and partially observable planning problems and identified the **belief state**—the set of actual states the agent might be in—as a key concept for describing and calculating solutions. In POMDPs, the belief state b becomes a *probability distribution* over all possible states, just as in Chapter 15. For example, the initial

[3] As with the reward function for MDPs, the sensor model can also depend on the action and outcome state, but again this change is not fundamental.

belief state for the 4×3 POMDP could be the uniform distribution over the nine nonterminal states, i.e., $\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$. We write $b(s)$ for the probability assigned to the actual state s by belief state b. The agent can calculate its current belief state as the conditional probability distribution over the actual states given the sequence of percepts and actions so far. This is essentially the **filtering** task described in Chapter 15. The basic recursive filtering equation (15.5 on page 572) shows how to calculate the new belief state from the previous belief state and the new evidence. For POMDPs, we also have an action to consider, but the result is essentially the same. If $b(s)$ was the previous belief state, and the agent does action a and then perceives evidence e, then the new belief state is given by

$$b'(s') = \alpha\, P(e \mid s') \sum_s P(s' \mid s, a) b(s) \ ,$$

where α is a normalizing constant that makes the belief state sum to 1. By analogy with the update operator for filtering (page 572), we can write this as

$$b' = \text{FORWARD}(b, a, e) \ . \tag{17.11}$$

In the 4×3 POMDP, suppose the agent moves *Left* and its sensor reports 1 adjacent wall; then it's quite likely (although not guaranteed, because both the motion and the sensor are noisy) that the agent is now in (3,1). Exercise 17.13 asks you to calculate the exact probability values for the new belief state.

The fundamental insight required to understand POMDPs is this: *the optimal action depends only on the agent's current belief state.* That is, the optimal policy can be described by a mapping $\pi^*(b)$ from belief states to actions. It does *not* depend on the *actual* state the agent is in. This is a good thing, because the agent does not know its actual state; all it knows is the belief state. Hence, the decision cycle of a POMDP agent can be broken down into the following three steps:

1. Given the current belief state b, execute the action $a = \pi^*(b)$.
2. Receive percept e.
3. Set the current belief state to $\text{FORWARD}(b, a, e)$ and repeat.

Now we can think of POMDPs as requiring a search in belief-state space, just like the methods for sensorless and contingency problems in Chapter 4. The main difference is that the POMDP belief-state space is *continuous*, because a POMDP belief state is a probability distribution. For example, a belief state for the 4×3 world is a point in an 11-dimensional continuous space. An action changes the belief state, not just the physical state. Hence, the action is evaluated at least in part according to the information the agent acquires as a result. POMDPs therefore include the value of information (Section 16.6) as one component of the decision problem.

Let's look more carefully at the outcome of actions. In particular, let's calculate the probability that an agent in belief state b reaches belief state b' after executing action a. Now, if we knew the action *and the subsequent percept*, then Equation (17.11) would provide a *deterministic* update to the belief state: $b' = \text{FORWARD}(b, a, e)$. Of course, the subsequent percept is not yet known, so the agent might arrive in one of several possible belief states b', depending on the percept that is received. The probability of perceiving e, given that a was

performed starting in belief state b, is given by summing over all the actual states s' that the agent might reach:

$$\begin{aligned} P(e|a,b) &= \sum_{s'} P(e|a,s',b)P(s'|a,b) \\ &= \sum_{s'} P(e \mid s')P(s'|a,b) \\ &= \sum_{s'} P(e \mid s') \sum_{s} P(s' \mid s,a)b(s) \ . \end{aligned}$$

Let us write the probability of reaching b' from b, given action a, as $P(b' \mid b,a))$. Then that gives us

$$\begin{aligned} P(b' \mid b,a) &= P(b'|a,b) = \sum_{e} P(b'|e,a,b)P(e|a,b) \\ &= \sum_{e} P(b'|e,a,b) \sum_{s'} P(e \mid s') \sum_{s} P(s' \mid s,a)b(s) \ , \end{aligned} \tag{17.12}$$

where $P(b'|e,a,b)$ is 1 if $b' = \text{FORWARD}(b,a,e)$ and 0 otherwise.

Equation (17.12) can be viewed as defining a transition model for the belief-state space. We can also define a reward function for belief states (i.e., the expected reward for the actual states the agent might be in):

$$\rho(b) = \sum_{s} b(s)R(s) \ .$$

Together, $P(b' \mid b,a)$ and $\rho(b)$ define an *observable* MDP on the space of belief states. Furthermore, it can be shown that an optimal policy for this MDP, $\pi^*(b)$, is also an optimal policy for the original POMDP. In other words, *solving a POMDP on a physical state space can be reduced to solving an MDP on the corresponding belief-state space.* This fact is perhaps less surprising if we remember that the belief state is always observable to the agent, by definition.

Notice that, although we have reduced POMDPs to MDPs, the MDP we obtain has a continuous (and usually high-dimensional) state space. None of the MDP algorithms described in Sections 17.2 and 17.3 applies directly to such MDPs. The next two subsections describe a value iteration algorithm designed specifically for POMDPs and an online decision-making algorithm, similar to those developed for games in Chapter 5.

17.4.2 Value iteration for POMDPs

Section 17.2 described a value iteration algorithm that computed one utility value for each state. With infinitely many belief states, we need to be more creative. Consider an optimal policy π^* and its application in a specific belief state b: the policy generates an action, then, for each subsequent percept, the belief state is updated and a new action is generated, and so on. For this specific b, therefore, the policy is exactly equivalent to a **conditional plan**, as defined in Chapter 4 for nondeterministic and partially observable problems. Instead of thinking about policies, let us think about conditional plans and how the expected utility of executing a fixed conditional plan varies with the initial belief state. We make two observations:

1. Let the utility of executing a *fixed* conditional plan p starting in physical state s be $\alpha_p(s)$. Then the expected utility of executing p in belief state b is just $\sum_s b(s)\alpha_p(s)$, or $b \cdot \alpha_p$ if we think of them both as vectors. Hence, the expected utility of a fixed conditional plan varies *linearly* with b; that is, it corresponds to a hyperplane in belief space.
2. At any given belief state b, the optimal policy will choose to execute the conditional plan with highest expected utility; and the expected utility of b under the optimal policy is just the utility of that conditional plan:

$$U(b) = U^{\pi^*}(b) = \max_p b \cdot \alpha_p \ .$$

   If the optimal policy π^* chooses to execute p starting at b, then it is reasonable to expect that it might choose to execute p in belief states that are very close to b; in fact, if we bound the depth of the conditional plans, then there are only finitely many such plans and the continuous space of belief states will generally be divided into *regions*, each corresponding to a particular conditional plan that is optimal in that region.

From these two observations, we see that the utility function $U(b)$ on belief states, being the maximum of a collection of hyperplanes, will be *piecewise linear* and *convex*.

To illustrate this, we use a simple two-state world. The states are labeled 0 and 1, with $R(0) = 0$ and $R(1) = 1$. There are two actions: *Stay* stays put with probability 0.9 and *Go* switches to the other state with probability 0.9. For now we will assume the discount factor $\gamma = 1$. The sensor reports the correct state with probability 0.6. Obviously, the agent should *Stay* when it thinks it's in state 1 and *Go* when it thinks it's in state 0.

The advantage of a two-state world is that the belief space can be viewed as one-dimensional, because the two probabilities must sum to 1. In Figure 17.8(a), the x-axis represents the belief state, defined by $b(1)$, the probability of being in state 1. Now let us consider the one-step plans $[Stay]$ and $[Go]$, each of which receives the reward for the current state followed by the (discounted) reward for the state reached after the action:

$$\begin{aligned}
\alpha_{[Stay]}(0) &= R(0) + \gamma(0.9R(0) + 0.1R(1)) = 0.1 \\
\alpha_{[Stay]}(1) &= R(1) + \gamma(0.9R(1) + 0.1R(0)) = 1.9 \\
\alpha_{[Go]}(0) &= R(0) + \gamma(0.9R(1) + 0.1R(0)) = 0.9 \\
\alpha_{[Go]}(1) &= R(1) + \gamma(0.9R(0) + 0.1R(1)) = 1.1
\end{aligned}$$

The hyperplanes (lines, in this case) for $b \cdot \alpha_{[Stay]}$ and $b \cdot \alpha_{[Go]}$ are shown in Figure 17.8(a) and their maximum is shown in bold. The bold line therefore represents the utility function for the finite-horizon problem that allows just one action, and in each "piece" of the piecewise linear utility function the optimal action is the first action of the corresponding conditional plan. In this case, the optimal one-step policy is to *Stay* when $b(1) > 0.5$ and *Go* otherwise.

Once we have utilities $\alpha_p(s)$ for all the conditional plans p of depth 1 in each physical state s, we can compute the utilities for conditional plans of depth 2 by considering each possible first action, each possible subsequent percept, and then each way of choosing a depth-1 plan to execute for each percept:

$[Stay;\ \textbf{if}\ Percept = 0\ \textbf{then}\ Stay\ \textbf{else}\ Stay]$

$[Stay;\ \textbf{if}\ Percept = 0\ \textbf{then}\ Stay\ \textbf{else}\ Go] \ldots$

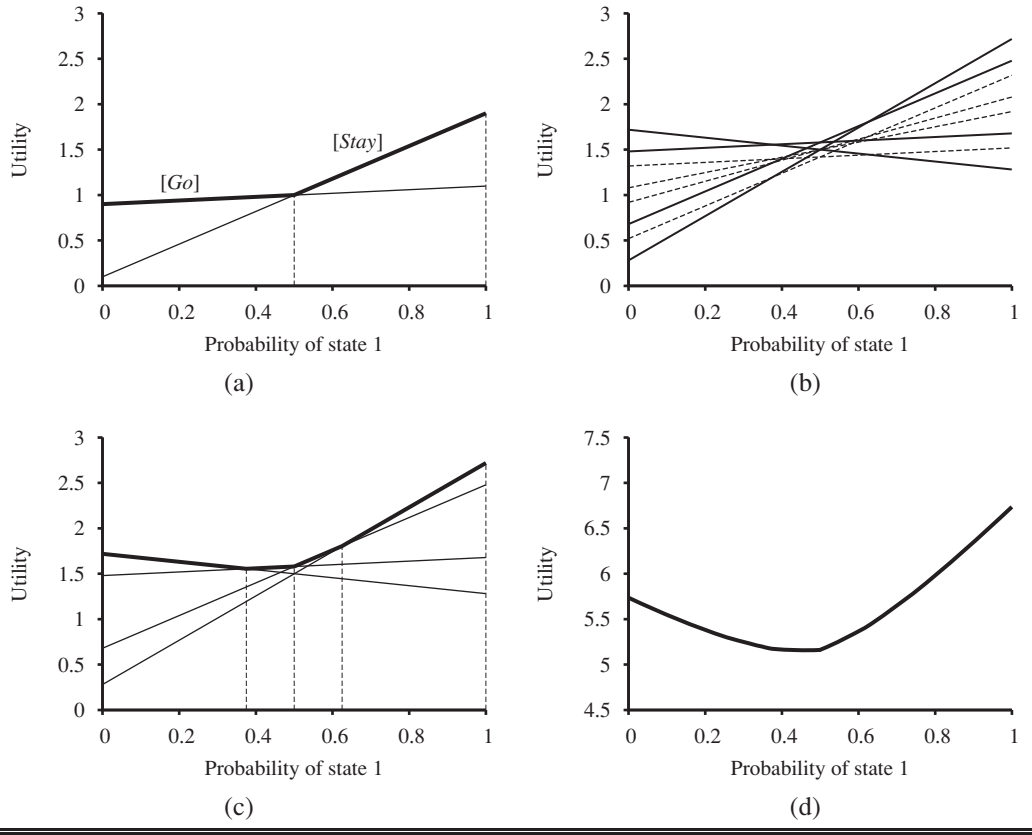


Figure 17.8 (a) Utility of two one-step plans as a function of the initial belief state $b(1)$ for the two-state world, with the corresponding utility function shown in bold. (b) Utilities for 8 distinct two-step plans. (c) Utilities for four undominated two-step plans. (d) Utility function for optimal eight-step plans.

There are eight distinct depth-2 plans in all, and their utilities are shown in Figure 17.8(b). Notice that four of the plans, shown as dashed lines, are suboptimal across the entire belief space—we say these plans are **dominated**, and they need not be considered further. There are four undominated plans, each of which is optimal in a specific region, as shown in Figure 17.8(c). The regions partition the belief-state space.

DOMINATED PLAN

We repeat the process for depth 3, and so on. In general, let p be a depth-d conditional plan whose initial action is a and whose depth-$d-1$ subplan for percept e is $p.e$; then

$$\alpha_p(s) = R(s) + \gamma \left(\sum_{s'} P(s' \mid s, a) \sum_{e} P(e \mid s') \alpha_{p.e}(s') \right) . \tag{17.13}$$

This recursion naturally gives us a value iteration algorithm, which is sketched in Figure 17.9. The structure of the algorithm and its error analysis are similar to those of the basic value iteration algorithm in Figure 17.4 on page 653; the main difference is that instead of computing one utility number for each state, POMDP-VALUE-ITERATION maintains a collection of

function POMDP-VALUE-ITERATION($pomdp, \epsilon$) **returns** a utility function
  inputs: $pomdp$, a POMDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    sensor model $P(e \mid s)$, rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', sets of plans p with associated utility vectors α_p

  $U' \leftarrow$ a set containing just the empty plan $[\,]$, with $\alpha_{[\,]}(s) = R(s)$
  repeat
    $U \leftarrow U'$
    $U' \leftarrow$ the set of all plans consisting of an action and, for each possible next percept,
      a plan in U with utility vectors computed according to Equation (17.13)
    $U' \leftarrow$ REMOVE-DOMINATED-PLANS(U')
  until MAX-DIFFERENCE(U, U') $< \epsilon(1-\gamma)/\gamma$
  return U

Figure 17.9 A high-level sketch of the value iteration algorithm for POMDPs. The REMOVE-DOMINATED-PLANS step and MAX-DIFFERENCE test are typically implemented as linear programs.

undominated plans with their utility hyperplanes. The algorithm's complexity depends primarily on how many plans get generated. Given $|A|$ actions and $|E|$ possible observations, it is easy to show that there are $|A|^{O(|E|^{d-1})}$ distinct depth-d plans. Even for the lowly two-state world with $d = 8$, the exact number is 2^{255}. The elimination of dominated plans is essential for reducing this doubly exponential growth: the number of undominated plans with $d = 8$ is just 144. The utility function for these 144 plans is shown in Figure 17.8(d).

Notice that even though state 0 has lower utility than state 1, the intermediate belief states have even lower utility because the agent lacks the information needed to choose a good action. This is why information has value in the sense defined in Section 16.6 and optimal policies in POMDPs often include information-gathering actions.

Given such a utility function, an executable policy can be extracted by looking at which hyperplane is optimal at any given belief state b and executing the first action of the corresponding plan. In Figure 17.8(d), the corresponding optimal policy is still the same as for depth-1 plans: *Stay* when $b(1) > 0.5$ and *Go* otherwise.

In practice, the value iteration algorithm in Figure 17.9 is hopelessly inefficient for larger problems—even the 4×3 POMDP is too hard. The main reason is that, given n conditional plans at level d, the algorithm constructs $|A| \cdot n^{|E|}$ conditional plans at level $d+1$ before eliminating the dominated ones. Since the 1970s, when this algorithm was developed, there have been several advances including more efficient forms of value iteration and various kinds of policy iteration algorithms. Some of these are discussed in the notes at the end of the chapter. For general POMDPs, however, finding optimal policies is very difficult (PSPACE-hard, in fact—i.e., very hard indeed). Problems with a few dozen states are often infeasible. The next section describes a different, approximate method for solving POMDPs, one based on look-ahead search.

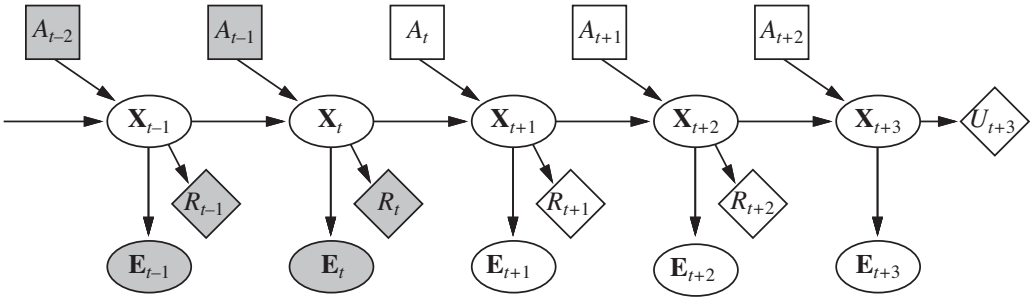


Figure 17.10 The generic structure of a dynamic decision network. Variables with known values are shaded. The current time is t and the agent must decide what to do—that is, choose a value for A_t. The network has been unrolled into the future for three steps and represents future rewards, as well as the utility of the state at the look-ahead horizon.

17.4.3 Online agents for POMDPs

In this section, we outline a simple approach to agent design for partially observable, stochastic environments. The basic elements of the design are already familiar:

- The transition and sensor models are represented by a **dynamic Bayesian network** (DBN), as described in Chapter 15.
- The dynamic Bayesian network is extended with decision and utility nodes, as used in **decision networks** in Chapter 16. The resulting model is called a **dynamic decision network**, or DDN.

DYNAMIC DECISION NETWORK

- A filtering algorithm is used to incorporate each new percept and action and to update the belief state representation.
- Decisions are made by projecting forward possible action sequences and choosing the best one.

DBNs are **factored representations** in the terminology of Chapter 2; they typically have an exponential complexity advantage over atomic representations and can model quite substantial real-world problems. The agent design is therefore a practical implementation of the **utility-based agent** sketched in Chapter 2.

In the DBN, the single state S_t becomes a set of state variables $\mathbf{X}_t$, and there may be multiple evidence variables $\mathbf{E}_t$. We will use A_t to refer to the action at time t, so the transition model becomes $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{X}_t, A_t)$ and the sensor model becomes $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$. We will use R_t to refer to the reward received at time t and U_t to refer to the utility of the state at time t. (Both of these are random variables.) With this notation, a dynamic decision network looks like the one shown in Figure 17.10.

Dynamic decision networks can be used as inputs for any POMDP algorithm, including those for value and policy iteration methods. In this section, we focus on look-ahead methods that project action sequences forward from the current belief state in much the same way as do the game-playing algorithms of Chapter 5. The network in Figure 17.10 has been projected three steps into the future; the current and future decisions A and the future observations

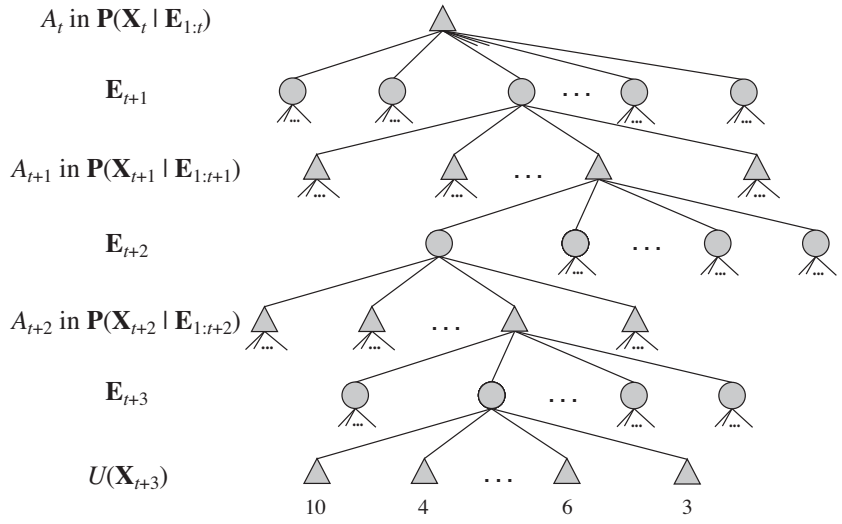


Figure 17.11 Part of the look-ahead solution of the DDN in Figure 17.10. Each decision will be taken in the belief state indicated.

$\mathbf{E}$ and rewards R are all unknown. Notice that the network includes nodes for the *rewards* for $\mathbf{X}_{t+1}$ and $\mathbf{X}_{t+2}$, but the *utility* for $\mathbf{X}_{t+3}$. This is because the agent must maximize the (discounted) sum of all future rewards, and $U(\mathbf{X}_{t+3})$ represents the reward for $\mathbf{X}_{t+3}$ and all subsequent rewards. As in Chapter 5, we assume that U is available only in some approximate form: if exact utility values were available, look-ahead beyond depth 1 would be unnecessary.

Figure 17.11 shows part of the search tree corresponding to the three-step look-ahead DDN in Figure 17.10. Each of the triangular nodes is a belief state in which the agent makes a decision A_{t+i} for $i = 0, 1, 2, \ldots$. The round (chance) nodes correspond to choices by the environment, namely, what evidence $\mathbf{E}_{t+i}$ arrives. Notice that there are no chance nodes corresponding to the action outcomes; this is because the belief-state update for an action is deterministic regardless of the actual outcome.

The belief state at each triangular node can be computed by applying a filtering algorithm to the sequence of percepts and actions leading to it. In this way, the algorithm takes into account the fact that, for decision A_{t+i}, the agent *will* have available percepts $\mathbf{E}_{t+1}, \ldots, \mathbf{E}_{t+i}$, even though at time t it does not know what those percepts will be. In this way, a decision-theoretic agent automatically takes into account the value of information and will execute information-gathering actions where appropriate.

A decision can be extracted from the search tree by backing up the utility values from the leaves, taking an average at the chance nodes and taking the maximum at the decision nodes. This is similar to the EXPECTIMINIMAX algorithm for game trees with chance nodes, except that (1) there can also be rewards at non-leaf states and (2) the decision nodes correspond to belief states rather than actual states. The time complexity of an exhaustive search to depth d is $O(|A|^d \cdot |\mathbf{E}|^d)$, where $|A|$ is the number of available actions and $|\mathbf{E}|$ is the number of possible percepts. (Notice that this is far less than the number of depth-d conditional

plans generated by value iteration.) For problems in which the discount factor γ is not too close to 1, a shallow search is often good enough to give near-optimal decisions. It is also possible to approximate the averaging step at the chance nodes, by sampling from the set of possible percepts instead of summing over all possible percepts. There are various other ways of finding good approximate solutions quickly, but we defer them to Chapter 21.

Decision-theoretic agents based on dynamic decision networks have a number of advantages compared with other, simpler agent designs presented in earlier chapters. In particular, they handle partially observable, uncertain environments and can easily revise their "plans" to handle unexpected evidence. With appropriate sensor models, they can handle sensor failure and can plan to gather information. They exhibit "graceful degradation" under time pressure and in complex environments, using various approximation techniques. So what is missing? One defect of our DDN-based algorithm is its reliance on forward search through state space, rather than using the hierarchical and other advanced planning techniques described in Chapter 11. There have been attempts to extend these techniques into the probabilistic domain, but so far they have proved to be inefficient. A second, related problem is the basically propositional nature of the DDN language. We would like to be able to extend some of the ideas for first-order probabilistic languages to the problem of decision making. Current research has shown that this extension is possible and has significant benefits, as discussed in the notes at the end of the chapter.

17.5 Decisions with Multiple Agents: Game Theory

This chapter has concentrated on making decisions in uncertain environments. But what if the uncertainty is due to other agents and the decisions they make? And what if the decisions of those agents are in turn influenced by our decisions? We addressed this question once before, when we studied games in Chapter 5. There, however, we were primarily concerned with turn-taking games in fully observable environments, for which minimax search can be used to find optimal moves. In this section we study the aspects of **game theory** that analyze games with simultaneous moves and other sources of partial observability. (Game theorists use the terms **perfect information** and **imperfect information** rather than fully and partially observable.) Game theory can be used in at least two ways:

GAME THEORY

1. **Agent design**: Game theory can analyze the agent's decisions and compute the expected utility for each decision (under the assumption that other agents are acting optimally according to game theory). For example, in the game **two-finger Morra**, two players, O and E, simultaneously display one or two fingers. Let the total number of fingers be f. If f is odd, O collects f dollars from E; and if f is even, E collects f dollars from O. Game theory can determine the best strategy against a rational player and the expected return for each player.[4]

[4] Morra is a recreational version of an **inspection game**. In such games, an inspector chooses a day to inspect a facility (such as a restaurant or a biological weapons plant), and the facility operator chooses a day to hide all the nasty stuff. The inspector wins if the days are different, and the facility operator wins if they are the same.

2. **Mechanism design**: When an environment is inhabited by many agents, it might be possible to define the rules of the environment (i.e., the game that the agents must play) so that the collective good of all agents is maximized when each agent adopts the game-theoretic solution that maximizes its own utility. For example, game theory can help design the protocols for a collection of Internet traffic routers so that each router has an incentive to act in such a way that global throughput is maximized. Mechanism design can also be used to construct intelligent **multiagent systems** that solve complex problems in a distributed fashion.

17.5.1 Single-move games

We start by considering a restricted set of games: ones where all players take action simultaneously and the result of the game is based on this single set of actions. (Actually, it is not crucial that the actions take place at exactly the same time; what matters is that no player has knowledge of the other players' choices.) The restriction to a single move (and the very use of the word "game") might make this seem trivial, but in fact, game theory is serious business. It is used in decision-making situations including the auctioning of oil drilling rights and wireless frequency spectrum rights, bankruptcy proceedings, product development and pricing decisions, and national defense—situations involving billions of dollars and hundreds of thousands of lives. A single-move game is defined by three components:

PLAYER
- **Players** or agents who will be making decisions. Two-player games have received the most attention, although n-player games for $n > 2$ are also common. We give players capitalized names, like *Alice* and *Bob* or O and E.

ACTION
- **Actions** that the players can choose. We will give actions lowercase names, like one or $testify$. The players may or may not have the same set of actions available.

PAYOFF FUNCTION
- A **payoff function** that gives the utility to each player for each combination of actions by all the players. For single-move games the payoff function can be represented by a matrix, a representation known as the **strategic form** (also called **normal form**). The payoff matrix for two-finger Morra is as follows:

STRATEGIC FORM

	O: one	*O: two*
E: one	$E = +2, O = -2$	$E = -3, O = +3$
E: two	$E = -3, O = +3$	$E = +4, O = -4$

For example, the lower-right corner shows that when player O chooses action *two* and E also chooses *two*, the payoff is +4 for E and -4 for O.

STRATEGY
PURE STRATEGY
MIXED STRATEGY
STRATEGY PROFILE
OUTCOME

Each player in a game must adopt and then execute a **strategy** (which is the name used in game theory for a *policy*). A **pure strategy** is a deterministic policy; for a single-move game, a pure strategy is just a single action. For many games an agent can do better with a **mixed strategy**, which is a randomized policy that selects actions according to a probability distribution. The mixed strategy that chooses action a with probability p and action b otherwise is written $[p\colon a; (1-p)\colon b]$. For example, a mixed strategy for two-finger Morra might be $[0.5\colon one; 0.5\colon two]$. A **strategy profile** is an assignment of a strategy to each player; given the strategy profile, the game's **outcome** is a numeric value for each player.

SOLUTION

A **solution** to a game is a strategy profile in which each player adopts a rational strategy. We will see that the most important issue in game theory is to define what "rational" means when each agent chooses only part of the strategy profile that determines the outcome. It is important to realize that outcomes are actual results of playing a game, while solutions are theoretical constructs used to analyze a game. We will see that some games have a solution only in mixed strategies. But that does not mean that a player must literally be adopting a mixed strategy to be rational.

PRISONER'S DILEMMA

Consider the following story: Two alleged burglars, Alice and Bob, are caught red-handed near the scene of a burglary and are interrogated separately. A prosecutor offers each a deal: if you testify against your partner as the leader of a burglary ring, you'll go free for being the cooperative one, while your partner will serve 10 years in prison. However, if you both testify against each other, you'll both get 5 years. Alice and Bob also know that if both refuse to testify they will serve only 1 year each for the lesser charge of possessing stolen property. Now Alice and Bob face the so-called **prisoner's dilemma**: should they testify or refuse? Being rational agents, Alice and Bob each want to maximize their own expected utility. Let's assume that Alice is callously unconcerned about her partner's fate, so her utility decreases in proportion to the number of years she will spend in prison, regardless of what happens to Bob. Bob feels exactly the same way. To help reach a rational decision, they both construct the following payoff matrix:

	$Alice$:$testify$	$Alice$:$refuse$
Bob:$testify$	$A = -5, B = -5$	$A = -10, B = 0$
Bob:$refuse$	$A = 0, B = -10$	$A = -1, B = -1$

Alice analyzes the payoff matrix as follows: "Suppose Bob testifies. Then I get 5 years if I testify and 10 years if I don't, so in that case testifying is better. On the other hand, if Bob refuses, then I get 0 years if I testify and 1 year if I refuse, so in that case as well testifying is better. So in either case, it's better for me to testify, so that's what I must do."

DOMINANT STRATEGY
STRONG DOMINATION
WEAK DOMINATION

Alice has discovered that $testify$ is a **dominant strategy** for the game. We say that a strategy s for player p **strongly dominates** strategy s' if the outcome for s is better for p than the outcome for s', for every choice of strategies by the other player(s). Strategy s **weakly dominates** s' if s is better than s' on at least one strategy profile and no worse on any other. A dominant strategy is a strategy that dominates all others. It is irrational to play a dominated strategy, and irrational not to play a dominant strategy if one exists. Being rational, Alice chooses the dominant strategy. We need just a bit more terminology: we say that an outcome is **Pareto optimal**[5] if there is no other outcome that all players would prefer. An outcome is **Pareto dominated** by another outcome if all players would prefer the other outcome.

PARETO OPTIMAL
PARETO DOMINATED

If Alice is clever as well as rational, she will continue to reason as follows: Bob's dominant strategy is also to testify. Therefore, he will testify and we will both get five years. When each player has a dominant strategy, the combination of those strategies is called a **dominant strategy equilibrium**. In general, a strategy profile forms an **equilibrium** if no player can benefit by switching strategies, given that every other player sticks with the same

DOMINANT STRATEGY EQUILIBRIUM
EQUILIBRIUM

[5] Pareto optimality is named after the economist Vilfredo Pareto (1848–1923).

strategy. An equilibrium is essentially a **local optimum** in the space of policies; it is the top of a peak that slopes downward along every dimension, where a dimension corresponds to a player's strategy choices.

NASH EQUILIBRIUM

The mathematician John Nash (1928–) proved that *every game has at least one equilibrium.* The general concept of equilibrium is now called **Nash equilibrium** in his honor. Clearly, a dominant strategy equilibrium is a Nash equilibrium (Exercise 17.16), but some games have Nash equilibria but no dominant strategies.

The *dilemma* in the prisoner's dilemma is that the equilibrium outcome is worse for both players than the outcome they would get if they both refused to testify. In other words, $(testify, testify)$ is Pareto dominated by the (-1, -1) outcome of $(refuse, refuse)$. Is there any way for Alice and Bob to arrive at the (-1, -1) outcome? It is certainly an *allowable* option for both of them to refuse to testify, but is is hard to see how rational agents can get there, given the definition of the game. Either player contemplating playing $refuse$ will realize that he or she would do better by playing $testify$. That is the attractive power of an equilibrium point. Game theorists agree that being a Nash equilibrium is a necessary condition for being a solution—although they disagree whether it is a sufficient condition.

It is easy enough to get to the $(refuse, refuse)$ solution if we modify the game. For example, we could change to a **repeated game** in which the players know that they will meet again. Or the agents might have moral beliefs that encourage cooperation and fairness. That means they have a different utility function, necessitating a different payoff matrix, making it a different game. We will see later that agents with limited computational powers, rather than the ability to reason absolutely rationally, can reach non-equilibrium outcomes, as can an agent that knows that the other agent has limited rationality. In each case, we are considering a different game than the one described by the payoff matrix above.

Now let's look at a game that has no dominant strategy. Acme, a video game console manufacturer, has to decide whether its next game machine will use Blu-ray discs or DVDs. Meanwhile, the video game software producer Best needs to decide whether to produce its next game on Blu-ray or DVD. The profits for both will be positive if they agree and negative if they disagree, as shown in the following payoff matrix:

	$Acme$:*bluray*	$Acme$:*dvd*
$Best$:*bluray*	$A = +9, B = +9$	$A = -4, B = -1$
$Best$:*dvd*	$A = -3, B = -1$	$A = +5, B = +5$

There is no dominant strategy equilibrium for this game, but there are *two* Nash equilibria: (*bluray, bluray*) and (*dvd, dvd*). We know these are Nash equilibria because if either player unilaterally moves to a different strategy, that player will be worse off. Now the agents have a problem: *there are multiple acceptable solutions, but if each agent aims for a different solution, then both agents will suffer.* How can they agree on a solution? One answer is that both should choose the Pareto-optimal solution (*bluray, bluray*); that is, we can restrict the definition of "solution" to the unique Pareto-optimal Nash equilibrium *provided that one exists*. Every game has at least one Pareto-optimal solution, but a game might have several, or they might not be equilibrium points. For example, if (*bluray, bluray*) had payoff (5, 5), then there would be two equal Pareto-optimal equilibrium points. To choose between

them the agents can either guess or *communicate*, which can be done either by establishing a convention that orders the solutions before the game begins or by negotiating to reach a mutually beneficial solution during the game (which would mean including communicative actions as part of a sequential game). Communication thus arises in game theory for exactly the same reasons that it arose in multiagent planning in Section 11.4. Games in which players need to communicate like this are called **coordination games**.

COORDINATION GAME

A game can have more than one Nash equilibrium; how do we know that every game must have at least one? Some games have no *pure-strategy* Nash equilibria. Consider, for example, any pure-strategy profile for two-finger Morra (page 666). If the total number of fingers is even, then O will want to switch; on the other hand (so to speak), if the total is odd, then E will want to switch. Therefore, no pure strategy profile can be an equilibrium and we must look to mixed strategies instead.

But *which* mixed strategy? In 1928, von Neumann developed a method for finding the *optimal* mixed strategy for two-player, **zero-sum games**—games in which the sum of the payoffs is always zero.[6] Clearly, Morra is such a game. For two-player, zero-sum games, we know that the payoffs are equal and opposite, so we need consider the payoffs of only one player, who will be the maximizer (just as in Chapter 5). For Morra, we pick the even player E to be the maximizer, so we can define the payoff matrix by the values $U_E(e, o)$—the payoff to E if E does e and O does o. (For convenience we call player E "her" and O "him.") Von Neumann's method is called the the **maximin** technique, and it works as follows:

ZERO-SUM GAME

MAXIMIN

- Suppose we change the rules as follows: first E picks her strategy and reveals it to O. Then O picks his strategy, with knowledge of E's strategy. Finally, we evaluate the expected payoff of the game based on the chosen strategies. This gives us a turn-taking game to which we can apply the standard **minimax** algorithm from Chapter 5. Let's suppose this gives an outcome $U_{E,O}$. Clearly, this game favors O, so the true utility U of the original game (from E's point of view) is *at least* $U_{E,O}$. For example, if we just look at pure strategies, the minimax game tree has a root value of -3 (see Figure 17.12(a)), so we know that $U \geq -3$.
- Now suppose we change the rules to force O to reveal his strategy first, followed by E. Then the minimax value of this game is $U_{O,E}$, and because this game favors E we know that U is *at most* $U_{O,E}$. With pure strategies, the value is $+2$ (see Figure 17.12(b)), so we know $U \leq +2$.

Combining these two arguments, we see that the true utility U of the solution to the original game must satisfy

$$U_{E,O} \leq U \leq U_{O,E} \qquad \text{or in this case,} \qquad -3 \leq U \leq 2\,.$$

To pinpoint the value of U, we need to turn our analysis to mixed strategies. First, observe the following: *once the first player has revealed his or her strategy, the second player might as well choose a pure strategy.* The reason is simple: if the second player plays a mixed strategy, $[p\text{: } one; (1-p)\text{: } two]$, its expected utility is a linear combination $(p \cdot u_{one} + (1-p) \cdot u_{two})$ of

[6] or a constant—see page 162.

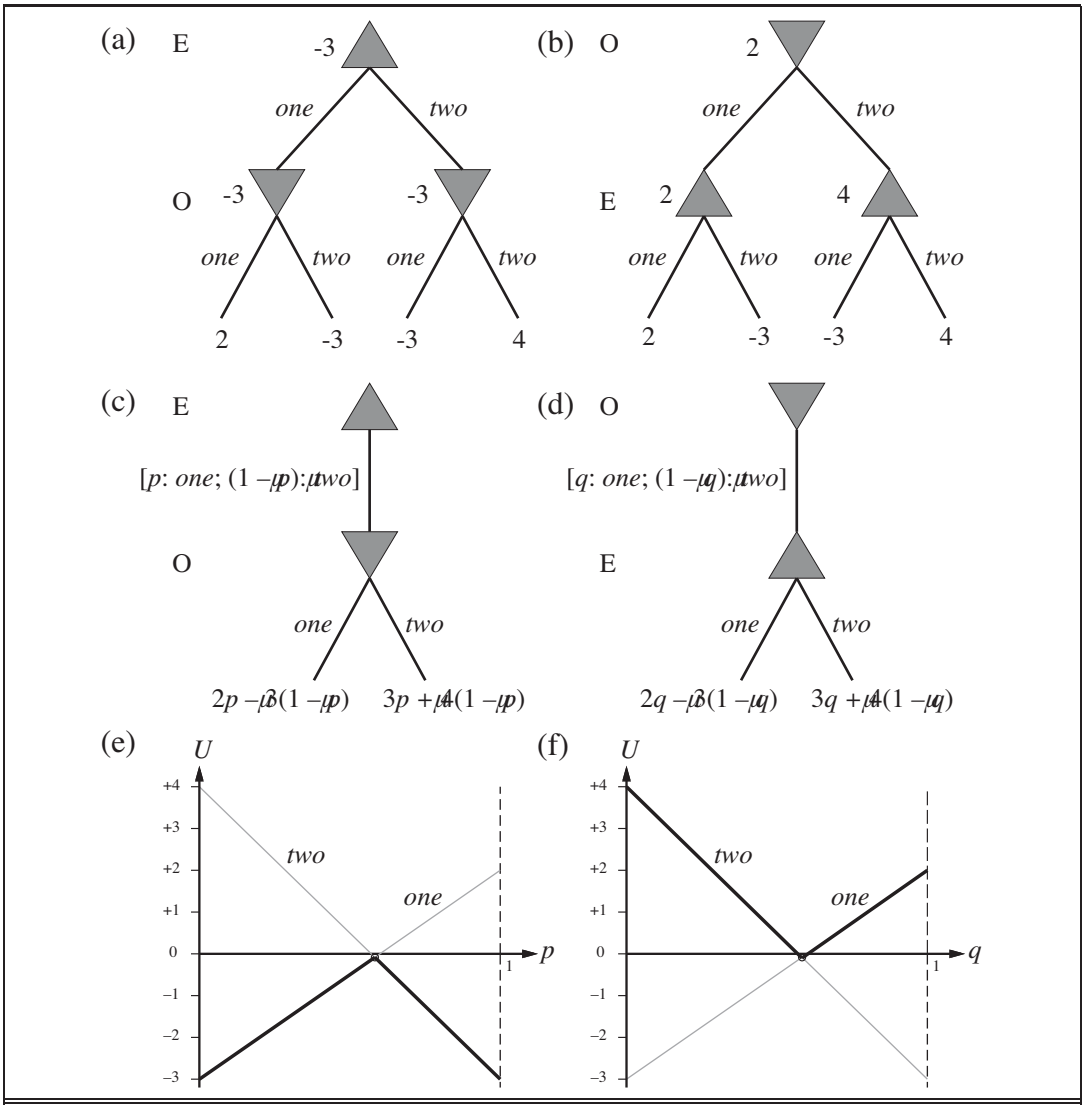


Figure 17.12 (a) and (b): Minimax game trees for two-finger Morra if the players take turns playing pure strategies. (c) and (d): Parameterized game trees where the first player plays a mixed strategy. The payoffs depend on the probability parameter (p or q) in the mixed strategy. (e) and (f): For any particular value of the probability parameter, the second player will choose the "better" of the two actions, so the value of the first player's mixed strategy is given by the heavy lines. The first player will choose the probability parameter for the mixed strategy at the intersection point.

the utilities of the pure strategies, u_{one} and u_{two}. This linear combination can never be better than the better of u_{one} and u_{two}, so the second player can just choose the better one.

With this observation in mind, the minimax trees can be thought of as having infinitely many branches at the root, corresponding to the infinitely many mixed strategies the first

player can choose. Each of these leads to a node with two branches corresponding to the pure strategies for the second player. We can depict these infinite trees finitely by having one "parameterized" choice at the root:

- If E chooses first, the situation is as shown in Figure 17.12(c). E chooses the strategy $[p\text{: } one; (1-p)\text{: } two]$ at the root, and then O chooses a pure strategy (and hence a move) given the value of p. If O chooses one, the expected payoff (to E) is $2p-3(1-p) = 5p-3$; if O chooses two, the expected payoff is $-3p + 4(1-p) = 4 - 7p$. We can draw these two payoffs as straight lines on a graph, where p ranges from 0 to 1 on the x-axis, as shown in Figure 17.12(e). O, the minimizer, will always choose the lower of the two lines, as shown by the heavy lines in the figure. Therefore, the best that E can do at the root is to choose p to be at the intersection point, which is where
$$5p - 3 = 4 - 7p \qquad \Rightarrow \qquad p = 7/12\ .$$
The utility for E at this point is $U_{E,O} = -1/12$.
- If O moves first, the situation is as shown in Figure 17.12(d). O chooses the strategy $[q\text{: } one; (1-q)\text{: } two]$ at the root, and then E chooses a move given the value of q. The payoffs are $2q-3(1-q) = 5q-3$ and $-3q+4(1-q) = 4-7q$.[7] Again, Figure 17.12(f) shows that the best O can do at the root is to choose the intersection point:
$$5q - 3 = 4 - 7q \qquad \Rightarrow \qquad q = 7/12\ .$$
The utility for E at this point is $U_{O,E} = -1/12$.

Now we know that the true utility of the original game lies between $-1/12$ and $-1/12$, that is, it is exactly $-1/12$! (The moral is that it is better to be O than E if you are playing this game.) Furthermore, the true utility is attained by the mixed strategy $[7/12\text{: } one; 5/12\text{: } two]$, which should be played by both players. This strategy is called the **maximin equilibrium** of the game, and is a Nash equilibrium. Note that each component strategy in an equilibrium mixed strategy has the same expected utility. In this case, both one and two have the same expected utility, $-1/12$, as the mixed strategy itself.

MAXIMIN EQUILIBRIUM

Our result for two-finger Morra is an example of the general result by von Neumann: *every two-player zero-sum game has a maximin equilibrium when you allow mixed strategies.* Furthermore, every Nash equilibrium in a zero-sum game is a maximin for both players. A player who adopts the maximin strategy has two guarantees: First, no other strategy can do better against an opponent who plays well (although some other strategies might be better at exploiting an opponent who makes irrational mistakes). Second, the player continues to do just as well even if the strategy is revealed to the opponent.

The general algorithm for finding maximin equilibria in zero-sum games is somewhat more involved than Figures 17.12(e) and (f) might suggest. When there are n possible actions, a mixed strategy is a point in n-dimensional space and the lines become hyperplanes. It's also possible for some pure strategies for the second player to be dominated by others, so that they are not optimal against *any* strategy for the first player. After removing all such strategies (which might have to be done repeatedly), the optimal choice at the root is the

[7] It is a coincidence that these equations are the same as those for p; the coincidence arises because $U_E(one, two) = U_E(two, one) = -3$. This also explains why the optimal strategy is the same for both players.

highest (or lowest) intersection point of the remaining hyperplanes. Finding this choice is an example of a **linear programming** problem: maximizing an objective function subject to linear constraints. Such problems can be solved by standard techniques in time polynomial in the number of actions (and in the number of bits used to specify the reward function, if you want to get technical).

The question remains, what should a rational agent actually *do* in playing a single game of Morra? The rational agent will have derived the fact that $[7/12: one; 5/12: two]$ is the maximin equilibrium strategy, and will assume that this is mutual knowledge with a rational opponent. The agent could use a 12-sided die or a random number generator to pick randomly according to this mixed strategy, in which case the expected payoff would be -1/12 for E. Or the agent could just decide to play one, or two. In either case, the expected payoff remains -1/12 for E. Curiously, unilaterally choosing a particular action does not harm one's expected payoff, but allowing the other agent to know that one has made such a unilateral decision *does* affect the expected payoff, because then the opponent can adjust his strategy accordingly.

Finding equilibria in non-zero-sum games is somewhat more complicated. The general approach has two steps: (1) Enumerate all possible subsets of actions that might form mixed strategies. For example, first try all strategy profiles where each player uses a single action, then those where each player uses either one or two actions, and so on. This is exponential in the number of actions, and so only applies to relatively small games. (2) For each strategy profile enumerated in (1), check to see if it is an equilibrium. This is done by solving a set of equations and inequalities that are similar to the ones used in the zero-sum case. For two players these equations are linear and can be solved with basic linear programming techniques, but for three or more players they are nonlinear and may be very difficult to solve.

17.5.2 Repeated games

REPEATED GAME

So far we have looked only at games that last a single move. The simplest kind of multiple-move game is the **repeated game**, in which players face the same choice repeatedly, but each time with knowledge of the history of all players' previous choices. A strategy profile for a repeated game specifies an action choice for each player at each time step for every possible history of previous choices. As with MDPs, payoffs are additive over time.

Let's consider the repeated version of the prisoner's dilemma. Will Alice and Bob work together and refuse to testify, knowing they will meet again? The answer depends on the details of the engagement. For example, suppose Alice and Bob know that they must play exactly 100 rounds of prisoner's dilemma. Then they both know that the 100th round will not be a repeated game—that is, its outcome can have no effect on future rounds—and therefore they will both choose the dominant strategy, $testify$, in that round. But once the 100th round is determined, the 99th round can have no effect on subsequent rounds, so it too will have a dominant strategy equilibrium at $(testify, testify)$. By induction, both players will choose $testify$ on every round, earning a total jail sentence of 500 years each.

We can get different solutions by changing the rules of the interaction. For example, suppose that after each round there is a 99% chance that the players will meet again. Then the expected number of rounds is still 100, but neither player knows for sure which round

will be the last. Under these conditions, more cooperative behavior is possible. For example, one equilibrium strategy is for each player to $refuse$ unless the other player has ever played $testify$. This strategy could be called **perpetual punishment**. Suppose both players have adopted this strategy, and this is mutual knowledge. Then as long as neither player has played $testify$, then at any point in time the expected future total payoff for each player is

PERPETUAL PUNISHMENT

$$\sum_{t=0}^{\infty} 0.99^t \cdot (-1) = -100 \ .$$

A player who deviates from the strategy and chooses $testify$ will gain a score of 0 rather than -1 on the very next move, but from then on both players will play $testify$ and the player's total expected future payoff becomes

$$0 + \sum_{t=1}^{\infty} 0.99^t \cdot (-5) = -495 \ .$$

Therefore, at every step, there is no incentive to deviate from $(refuse, refuse)$. Perpetual punishment is the "mutually assured destruction" strategy of the prisoner's dilemma: once either player decides to $testify$, it ensures that both players suffer a great deal. But it works as a deterrent only if the other player believes you have adopted this strategy—or at least that you might have adopted it.

TIT-FOR-TAT

Other strategies are more forgiving. The most famous, called **tit-for-tat**, calls for starting with $refuse$ and then echoing the other player's previous move on all subsequent moves. So Alice would refuse as long as Bob refuses and would testify the move after Bob testified, but would go back to refusing if Bob did. Although very simple, this strategy has proven to be highly robust and effective against a wide variety of strategies.

We can also get different solutions by changing the agents, rather than changing the rules of engagement. Suppose the agents are finite-state machines with n states and they are playing a game with $m > n$ total steps. The agents are thus incapable of representing the number of remaining steps, and must treat it as an unknown. Therefore, they cannot do the induction, and are free to arrive at the more favorable (*refuse, refuse*) equilibrium. In this case, ignorance *is* bliss—or rather, having your opponent believe that you are ignorant is bliss. Your success in these repeated games depends on the other player's *perception* of you as a bully or a simpleton, and not on your actual characteristics.

17.5.3 Sequential games

In the general case, a game consists of a sequence of turns that need not be all the same. Such games are best represented by a game tree, which game theorists call the **extensive form**. The tree includes all the same information we saw in Section 5.1: an initial state S_0, a function PLAYER(s) that tells which player has the move, a function ACTIONS(s) enumerating the possible actions, a function RESULT(s, a) that defines the transition to a new state, and a partial function UTILITY(s, p), which is defined only on terminal states, to give the payoff for each player.

EXTENSIVE FORM

To represent stochastic games, such as backgammon, we add a distinguished player, *chance*, that can take random actions. *Chance*'s "strategy" is part of the definition of the

game, specified as a probability distribution over actions (the other players get to choose their own strategy). To represent games with nondeterministic actions, such as billiards, we break the action into two pieces: the player's action itself has a deterministic result, and then *chance* has a turn to react to the action in its own capricious way. To represent simultaneous moves, as in the prisoner's dilemma or two-finger Morra, we impose an arbitrary order on the players, but we have the option of asserting that the earlier player's actions are not observable to the subsequent players: e.g., Alice must choose *refuse* or *testify* first, then Bob chooses, but Bob does not know what choice Alice made at that time (we can also represent the fact that the move is revealed later). However, we assume the players always remember all their *own* previous actions; this assumption is called **perfect recall**.

The key idea of extensive form that sets it apart from the game trees of Chapter 5 is the representation of partial observability. We saw in Section 5.6 that a player in a partially observable game such as Kriegspiel can create a game tree over the space of **belief states**. With that tree, we saw that in some cases a player can find a sequence of moves (a strategy) that leads to a forced checkmate regardless of what actual state we started in, and regardless of what strategy the opponent uses. However, the techniques of Chapter 5 could not tell a player what to do when there is no guaranteed checkmate. If the player's best strategy depends on the opponent's strategy and vice versa, then minimax (or alpha–beta) by itself cannot find a solution. The extensive form *does* allow us to find solutions because it represents the belief states (game theorists call them **information sets**) of *all* players at once. From that representation we can find equilibrium solutions, just as we did with normal-form games.

INFORMATION SETS

As a simple example of a sequential game, place two agents in the 4×3 world of Figure 17.1 and have them move simultaneously until one agent reaches an exit square, and gets the payoff for that square. If we specify that no movement occurs when the two agents try to move into the same square simultaneously (a common problem at many traffic intersections), then certain pure strategies can get stuck forever. Thus, agents need a mixed strategy to perform well in this game: randomly choose between moving ahead and staying put. This is exactly what is done to resolve packet collisions in Ethernet networks.

Next we'll consider a very simple variant of poker. The deck has only four cards, two aces and two kings. One card is dealt to each player. The first player then has the option to *raise* the stakes of the game from 1 point to 2, or to *check*. If player 1 checks, the game is over. If he raises, then player 2 has the option to *call*, accepting that the game is worth 2 points, or *fold*, conceding the 1 point. If the game does not end with a fold, then the payoff depends on the cards: it is zero for both players if they have the same card; otherwise the player with the king pays the stakes to the player with the ace.

The extensive-form tree for this game is shown in Figure 17.13. Nonterminal states are shown as circles, with the player to move inside the circle; player 0 is *chance*. Each action is depicted as an arrow with a label, corresponding to a *raise, check, call,* or *fold*, or, for *chance*, the four possible deals ("AK" means that player 1 gets an ace and player 2 a king). Terminal states are rectangles labeled by their payoff to player 1 and player 2. Information sets are shown as labeled dashed boxes; for example, $I_{1,1}$ is the information set where it is player 1's turn, and he knows he has an ace (but does not know what player 2 has). In information set $I_{2,1}$, it is player 2's turn and she knows that she has an ace and that player 1 has raised,

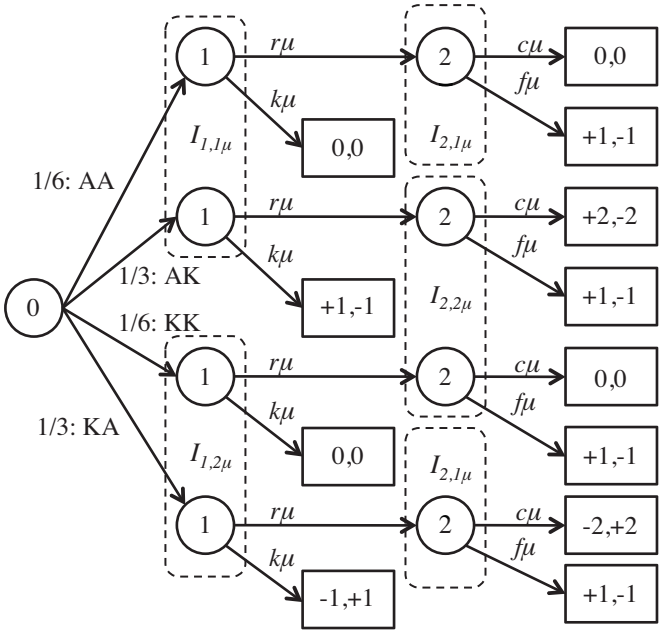


Figure 17.13 Extensive form of a simplified version of poker.

but does not know what card player 1 has. (Due to the limits of two-dimensional paper, this information set is shown as two boxes rather than one.)

One way to solve an extensive game is to convert it to a normal-form game. Recall that the normal form is a matrix, each row of which is labeled with a pure strategy for player 1, and each column by a pure strategy for player 2. In an extensive game a pure strategy for player *i* corresponds to an action for each information set involving that player. So in Figure 17.13, one pure strategy for player 1 is "raise when in $I_{1,1}$ (that is, when I have an ace), and check when in $I_{1,2}$ (when I have a king)." In the payoff matrix below, this strategy is called *rk*. Similarly, strategy *cf* for player 2 means "call when I have an ace and fold when I have a king." Since this is a zero-sum game, the matrix below gives only the payoff for player 1; player 2 always has the opposite payoff:

	2:*cc*	2:*cf*	2:*ff*	2:*fc*
1:*rr*	0	-1/6	1	7/6
1:*kr*	-1/3	-1/6	5/6	2/3
1:*rk*	1/3	**0**	1/6	1/2
1:*kk*	0	**0**	0	0

This game is so simple that it has two pure-strategy equilibria, shown in bold: *cf* for player 2 and *rk* or *kk* for player 1. But in general we can solve extensive games by converting to normal form and then finding a solution (usually a mixed strategy) using standard linear programming methods. That works in theory. But if a player has I information sets and a actions per set, then that player will have a^I pure strategies. In other words, the size of the normal-form matrix is exponential in the number of information sets, so in practice the

approach works only for very small game trees, on the order of a dozen states. A game like Texas hold'em poker has about 10^{18} states, making this approach completely infeasible.

What are the alternatives? In Chapter 5 we saw how alpha–beta search could handle games of perfect information with huge game trees by generating the tree incrementally, by pruning some branches, and by heuristically evaluating nonterminal nodes. But that approach does not work well for games with imperfect information, for two reasons: first, it is harder to prune, because we need to consider mixed strategies that combine multiple branches, not a pure strategy that always chooses the best branch. Second, it is harder to heuristically evaluate a nonterminal node, because we are dealing with information sets, not individual states.

SEQUENCE FORM

Koller *et al.* (1996) come to the rescue with an alternative representation of extensive games, called the **sequence form**, that is only linear in the size of the tree, rather than exponential. Rather than represent strategies, it represents paths through the tree; the number of paths is equal to the number of terminal nodes. Standard linear programming methods can again be applied to this representation. The resulting system can solve poker variants with 25,000 states in a minute or two. This is an exponential speedup over the normal-form approach, but still falls far short of handling full poker, with 10^{18} states.

ABSTRACTION

If we can't handle 10^{18} states, perhaps we can simplify the problem by changing the game to a simpler form. For example, if I hold an ace and am considering the possibility that the next card will give me a pair of aces, then I don't care about the suit of the next card; any suit will do equally well. This suggests forming an **abstraction** of the game, one in which suits are ignored. The resulting game tree will be smaller by a factor of $4! = 24$. Suppose I can solve this smaller game; how will the solution to that game relate to the original game? If no player is going for a flush (or bluffing so), then the suits don't matter to any player, and the solution for the abstraction will also be a solution for the original game. However, if any player is contemplating a flush, then the abstraction will be only an approximate solution (but it is possible to compute bounds on the error).

There are many opportunities for abstraction. For example, at the point in a game where each player has two cards, if I hold a pair of queens, then the other players' hands could be abstracted into three classes: *better* (only a pair of kings or a pair of aces), *same* (pair of queens) or *worse* (everything else). However, this abstraction might be too coarse. A better abstraction would divide *worse* into, say, *medium pair* (nines through jacks), *low pair*, and *no pair*. These examples are abstractions of states; it is also possible to abstract actions. For example, instead of having a bet action for each integer from 1 to 1000, we could restrict the bets to 10^0, 10^1, 10^2 and 10^3. Or we could cut out one of the rounds of betting altogether. We can also abstract over chance nodes, by considering only a subset of the possible deals. This is equivalent to the rollout technique used in Go programs. Putting all these abstractions together, we can reduce the 10^{18} states of poker to 10^7 states, a size that can be solved with current techniques.

Poker programs based on this approach can easily defeat novice and some experienced human players, but are not yet at the level of master players. Part of the problem is that the solution these programs approximate—the equilibrium solution—is optimal only against an opponent who also plays the equilibrium strategy. Against fallible human players it is important to be able to exploit an opponent's deviation from the equilibrium strategy. As

Gautam Rao (aka "The Count"), the world's leading online poker player, said (Billings *et al.*, 2003), "You have a very strong program. Once you add opponent modeling to it, it will kill everyone." However, good models of human fallability remain elusive.

In a sense, extensive game form is the one of the most complete representations we have seen so far: it can handle partially observable, multiagent, stochastic, sequential, dynamic environments—most of the hard cases from the list of environment properties on page 42. However, there are two limitations of game theory. First, it does not deal well with continuous states and actions (although there have been some extensions to the continuous case; for example, the theory of **Cournot competition** uses game theory to solve problems where two companies choose prices for their products from a continuous space). Second, game theory assumes the game is *known*. Parts of the game may be specified as unobservable to some of the players, but it must be known what parts are unobservable. In cases in which the players learn the unknown structure of the game over time, the model begins to break down. Let's examine each source of uncertainty, and whether each can be represented in game theory.

COURNOT COMPETITION

Actions: There is no easy way to represent a game where the players have to discover what actions are available. Consider the game between computer virus writers and security experts. Part of the problem is anticipating what action the virus writers will try next.

Strategies: Game theory is very good at representing the idea that the other players' strategies are initially unknown—as long as we assume all agents are rational. The theory itself does not say what to do when the other players are less than fully rational. The notion of a **Bayes–Nash equilibrium** partially addresses this point: it is an equilibrium with respect to a player's prior probability distribution over the other players' strategies—in other words, it expresses a player's beliefs about the other players' likely strategies.

BAYES–NASH EQUILIBRIUM

Chance: If a game depends on the roll of a die, it is easy enough to model a chance node with uniform distribution over the outcomes. But what if it is possible that the die is unfair? We can represent that with another chance node, higher up in the tree, with two branches for "die is fair" and "die is unfair," such that the corresponding nodes in each branch are in the same information set (that is, the players don't know if the die is fair or not). And what if we suspect the other opponent does know? Then we add *another* chance node, with one branch representing the case where the opponent does know, and one where he doesn't.

Utilities: What if we don't know our opponent's utilities? Again, that can be modeled with a chance node, such that the other agent knows its own utilities in each branch, but we don't. But what if we don't know our *own* utilities? For example, how do I know if it is rational to order the Chef's salad if I don't know how much I will like it? We can model that with yet another chance node specifying an unobservable "intrinsic quality" of the salad.

Thus, we see that game theory is good at representing most sources of uncertainty—but at the cost of doubling the size of the tree every time we add another node; a habit which quickly leads to intractably large trees. Because of these and other problems, game theory has been used primarily to *analyze* environments that are at equilibrium, rather than to *control* agents within an environment. Next we shall see how it can help *design* environments.

17.6 MECHANISM DESIGN

In the previous section, we asked, "Given a game, what is a rational strategy?" In this section, we ask, "Given that agents pick rational strategies, what game should we design?" More specifically, we would like to design a game whose solutions, consisting of each agent pursuing its own rational strategy, result in the maximization of some global utility function. This problem is called **mechanism design**, or sometimes **inverse game theory**. Mechanism design is a staple of economics and political science. Capitalism 101 says that if everyone tries to get rich, the total wealth of society will increase. But the examples we will discuss show that proper mechanism design is necessary to keep the invisible hand on track. For collections of agents, mechanism design allows us to construct smart systems out of a collection of more limited systems—even uncooperative systems—in much the same way that teams of humans can achieve goals beyond the reach of any individual.

MECHANISM DESIGN

Examples of mechanism design include auctioning off cheap airline tickets, routing TCP packets between computers, deciding how medical interns will be assigned to hospitals, and deciding how robotic soccer players will cooperate with their teammates. Mechanism design became more than an academic subject in the 1990s when several nations, faced with the problem of auctioning off licenses to broadcast in various frequency bands, lost hundreds of millions of dollars in potential revenue as a result of poor mechanism design. Formally, a **mechanism** consists of (1) a language for describing the set of allowable strategies that agents may adopt, (2) a distinguished agent, called the **center**, that collects reports of strategy choices from the agents in the game, and (3) an outcome rule, known to all agents, that the center uses to determine the payoffs to each agent, given their strategy choices.

MECHANISM

CENTER

17.6.1 Auctions

Let's consider **auctions** first. An auction is a mechanism for selling some goods to members of a pool of bidders. For simplicity, we concentrate on auctions with a single item for sale. Each bidder i has a utility value v_i for having the item. In some cases, each bidder has a **private value** for the item. For example, the first item sold on eBay was a broken laser pointer, which sold for \$14.83 to a collector of broken laser pointers. Thus, we know that the collector has $v_i \geq \$14.83$, but most other people would have $v_j \ll \$14.83$. In other cases, such as auctioning drilling rights for an oil tract, the item has a **common value**—the tract will produce some amount of money, X, and all bidders value a dollar equally—but there is uncertainty as to what the actual value of X is. Different bidders have different information, and hence different estimates of the item's true value. In either case, bidders end up with their own v_i. Given v_i, each bidder gets a chance, at the appropriate time or times in the auction, to make a bid b_i. The highest bid, b_{max} wins the item, but the price paid need not be b_{max}; that's part of the mechanism design.

AUCTION

The best-known auction mechanism is the **ascending-bid**,[8] or **English auction**, in which the center starts by asking for a minimum (or **reserve**) bid b_{min}. If some bidder is

ASCENDING-BID

ENGLISH AUCTION

[8] The word "auction" comes from the Latin *augere*, to increase.

willing to pay that amount, the center then asks for $b_{min} + d$, for some increment d, and continues up from there. The auction ends when nobody is willing to bid anymore; then the last bidder wins the item, paying the price he bid.

How do we know if this is a good mechanism? One goal is to maximize expected revenue for the seller. Another goal is to maximize a notion of global utility. These goals overlap to some extent, because one aspect of maximizing global utility is to ensure that the winner of the auction is the agent who values the item the most (and thus is willing to pay the most). We say an auction is **efficient** if the goods go to the agent who values them most. The ascending-bid auction is usually both efficient and revenue maximizing, but if the reserve price is set too high, the bidder who values it most may not bid, and if the reserve is set too low, the seller loses net revenue.

EFFICIENT

Probably the most important things that an auction mechanism can do is encourage a sufficient number of bidders to enter the game and discourage them from engaging in **collusion**. Collusion is an unfair or illegal agreement by two or more bidders to manipulate prices. It can happen in secret backroom deals or tacitly, within the rules of the mechanism.

COLLUSION

For example, in 1999, Germany auctioned ten blocks of cell-phone spectrum with a simultaneous auction (bids were taken on all ten blocks at the same time), using the rule that any bid must be a minimum of a 10% raise over the previous bid on a block. There were only two credible bidders, and the first, Mannesman, entered the bid of 20 million deutschmark on blocks 1-5 and 18.18 million on blocks 6-10. Why 18.18M? One of T-Mobile's managers said they "interpreted Mannesman's first bid as an offer." Both parties could compute that a 10% raise on 18.18M is 19.99M; thus Mannesman's bid was interpreted as saying "we can each get half the blocks for 20M; let's not spoil it by bidding the prices up higher." And in fact T-Mobile bid 20M on blocks 6-10 and that was the end of the bidding. The German government got less than they expected, because the two competitors were able to use the bidding mechanism to come to a tacit agreement on how not to compete. From the government's point of view, a better result could have been obtained by any of these changes to the mechanism: a higher reserve price; a sealed-bid first-price auction, so that the competitors could not communicate through their bids; or incentives to bring in a third bidder. Perhaps the 10% rule was an error in mechanism design, because it facilitated the precise signaling from Mannesman to T-Mobile.

In general, both the seller and the global utility function benefit if there are more bidders, although global utility can suffer if you count the cost of wasted time of bidders that have no chance of winning. One way to encourage more bidders is to make the mechanism easier for them. After all, if it requires too much research or computation on the part of the bidders, they may decide to take their money elsewhere. So it is desirable that the bidders have a **dominant strategy**. Recall that "dominant" means that the strategy works against all other strategies, which in turn means that an agent can adopt it without regard for the other strategies. An agent with a dominant strategy can just bid, without wasting time contemplating other agents' possible strategies. A mechanism where agents have a dominant strategy is called a **strategy-proof** mechanism. If, as is usually the case, that strategy involves the bidders revealing their true value, v_i, then it is called a **truth-revealing**, or **truthful**, auction; the term **incentive compatible** is also used. The **revelation principle** states that any mecha-

STRATEGY-PROOF

TRUTH-REVEALING

REVELATION PRINCIPLE

nism can be transformed into an equivalent truth-revealing mechanism, so part of mechanism design is finding these equivalent mechanisms.

It turns out that the ascending-bid auction has most of the desirable properties. The bidder with the highest value v_i gets the goods at a price of $b_o + d$, where b_o is the highest bid among all the other agents and d is the auctioneer's increment.[9] Bidders have a simple dominant strategy: keep bidding as long as the current cost is below your v_i. The mechanism is not quite truth-revealing, because the winning bidder reveals only that his $v_i \geq b_o + d$; we have a lower bound on v_i but not an exact amount.

A disadvantage (from the point of view of the seller) of the ascending-bid auction is that it can discourage competition. Suppose that in a bid for cell-phone spectrum there is one advantaged company that everyone agrees would be able to leverage existing customers and infrastructure, and thus can make a larger profit than anyone else. Potential competitors can see that they have no chance in an ascending-bid auction, because the advantaged company can always bid higher. Thus, the competitors may not enter at all, and the advantaged company ends up winning at the reserve price.

Another negative property of the English auction is its high communication costs. Either the auction takes place in one room or all bidders have to have high-speed, secure communication lines; in either case they have to have the time available to go through several rounds of bidding. An alternative mechanism, which requires much less communication, is the **sealed-bid auction**. Each bidder makes a single bid and communicates it to the auctioneer, without the other bidders seeing it. With this mechanism, there is no longer a simple dominant strategy. If your value is v_i and you believe that the maximum of all the other agents' bids will be b_o, then you should bid $b_o + \epsilon$, for some small ϵ, if that is less than v_i. Thus, your bid depends on your estimation of the other agents' bids, requiring you to do more work. Also, note that the agent with the highest v_i might not win the auction. This is offset by the fact that the auction is more competitive, reducing the bias toward an advantaged bidder.

SEALED-BID AUCTION

A small change in the mechanism for sealed-bid auctions produces the **sealed-bid second-price auction**, also known as a **Vickrey auction**.[10] In such auctions, the winner pays the price of the *second*-highest bid, b_o, rather than paying his own bid. This simple modification completely eliminates the complex deliberations required for standard (or **first-price**) sealed-bid auctions, because the dominant strategy is now simply to bid v_i; the mechanism is truth-revealing. Note that the utility of agent i in terms of his bid b_i, his value v_i, and the best bid among the other agents, b_o, is

SEALED-BID SECOND-PRICE AUCTION
VICKREY AUCTION

$$u_i = \begin{cases} (v_i - b_o) & \text{if } b_i > b_o \\ 0 & \text{otherwise.} \end{cases}$$

To see that $b_i = v_i$ is a dominant strategy, note that when $(v_i - b_o)$ is positive, any bid that wins the auction is optimal, and bidding v_i in particular wins the auction. On the other hand, when $(v_i - b_o)$ is negative, any bid that loses the auction is optimal, and bidding v_i in

[9] There is actually a small chance that the agent with highest v_i fails to get the goods, in the case in which $b_o < v_i < b_o + d$. The chance of this can be made arbitrarily small by decreasing the increment d.

[10] Named after William Vickrey (1914–1996), who won the 1996 Nobel Prize in economics for this work and died of a heart attack three days later.

particular loses the auction. So bidding v_i is optimal for all possible values of b_o, and in fact, v_i is the only bid that has this property. Because of its simplicity and the minimal computation requirements for both seller and bidders, the Vickrey auction is widely used in constructing distributed AI systems. Also, Internet search engines conduct over a billion auctions a day to sell advertisements along with their search results, and online auction sites handle $100 billion a year in goods, all using variants of the Vickrey auction. Note that the expected value to the seller is b_o, which is the same expected return as the limit of the English auction as the increment d goes to zero. This is actually a very general result: the **revenue equivalence theorem** states that, with a few minor caveats, any auction mechanism where risk-neutral bidders have values v_i known only to themselves (but know a probability distribution from which those values are sampled), will yield the same expected revenue. This principle means that the various mechanisms are not competing on the basis of revenue generation, but rather on other qualities.

REVENUE EQUIVALENCE THEOREM

Although the second-price auction is truth-revealing, it turns out that extending the idea to multiple goods and using a next-price auction is not truth-revealing. Many Internet search engines use a mechanism where they auction k slots for ads on a page. The highest bidder wins the top spot, the second highest gets the second spot, and so on. Each winner pays the price bid by the next-lower bidder, with the understanding that payment is made only if the searcher actually clicks on the ad. The top slots are considered more valuable because they are more likely to be noticed and clicked on. Imagine that three bidders, b_1, b_2 and b_3, have valuations for a click of $v_1 = 200, v_2 = 180$, and $v_3 = 100$, and that$k = 2$ slots are available, where it is known that the top spot is clicked on 5% of the time and the bottom spot 2%. If all bidders bid truthfully, then b_1 wins the top slot and pays 180, and has an expected return of $(200 - 180) \times 0.05 = 1$. The second slot goes to b_2. But b_1 can see that if she were to bid anything in the range 101–179, she would concede the top slot to b_2, win the second slot, and yield an expected return of $(200 - 100) \times .02 = 2$. Thus, b_1 can double her expected return by bidding less than her true value in this case. In general, bidders in this multislot auction must spend a lot of energy analyzing the bids of others to determine their best strategy; there is no simple dominant strategy. Aggarwal *et al.* (2006) show that there is a unique truthful auction mechanism for this multislot problem, in which the winner of slot j pays the full price for slot j just for those additional clicks that are available at slot j and not at slot $j + 1$. The winner pays the price for the lower slot for the remaining clicks. In our example, b_1 would bid 200 truthfully, and would pay 180 for the additional $.05 - .02 = .03$ clicks in the top slot, but would pay only the cost of the bottom slot, 100, for the remaining .02 clicks. Thus, the total return to b_1 would be $(200 - 180) \times .03 + (200 - 100) \times .02 = 2.6$.

Another example of where auctions can come into play within AI is when a collection of agents are deciding whether to cooperate on a joint plan. Hunsberger and Grosz (2000) show that this can be accomplished efficiently with an auction in which the agents bid for roles in the joint plan.

17.6.2 Common goods

Now let's consider another type of game, in which countries set their policy for controlling air pollution. Each country has a choice: they can reduce pollution at a cost of -10 points for implementing the necessary changes, or they can continue to pollute, which gives them a net utility of -5 (in added health costs, etc.) and also contributes -1 points to every other country (because the air is shared across countries). Clearly, the dominant strategy for each country is "continue to pollute," but if there are 100 countries and each follows this policy, then each country gets a total utility of -104, whereas if every country reduced pollution, they would each have a utility of -10. This situation is called the **tragedy of the commons**: if nobody has to pay for using a common resource, then it tends to be exploited in a way that leads to a lower total utility for all agents. It is similar to the prisoner's dilemma: there is another solution to the game that is better for all parties, but there appears to be no way for rational agents to arrive at that solution.

TRAGEDY OF THE COMMONS

The standard approach for dealing with the tragedy of the commons is to change the mechanism to one that charges each agent for using the commons. More generally, we need to ensure that all **externalities**—effects on global utility that are not recognized in the individual agents' transactions—are made explicit. Setting the prices correctly is the difficult part. In the limit, this approach amounts to creating a mechanism in which each agent is effectively required to maximize global utility, but can do so by making a local decision. For this example, a carbon tax would be an example of a mechanism that charges for use of the commons in a way that, if implemented well, maximizes global utility.

EXTERNALITIES

As a final example, consider the problem of allocating some common goods. Suppose a city decides it wants to install some free wireless Internet transceivers. However, the number of transceivers they can afford is less than the number of neighborhoods that want them. The city wants to allocate the goods efficiently, to the neighborhoods that would value them the most. That is, they want to maximize the global utility $V = \sum_i v_i$. The problem is that if they just ask each neighborhood council "how much do you value this free gift?" they would all have an incentive to lie, and report a high value. It turns out there is a mechanism, known as the **Vickrey-Clarke-Groves**, or **VCG**, mechanism, that makes it a dominant strategy for each agent to report its true utility and that achieves an efficient allocation of the goods. The trick is that each agent pays a tax equivalent to the loss in global utility that occurs because of the agent's presence in the game. The mechanism works like this:

VICKREY-CLARKE-GROVES

VCG

1. The center asks each agent to report its value for receiving an item. Call this b_i.
2. The center allocates the goods to a subset of the bidders. We call this subset A, and use the notation $b_i(A)$ to mean the result to i under this allocation: b_i if i is in A (that is, i is a winner), and 0 otherwise. The center chooses A to maximize total reported utility $B = \sum_i b_i(A)$.
3. The center calculates (for each i) the sum of the reported utilities for all the winners except i. We use the notation $B_{-i} = \sum_{j \neq i} b_j(A)$. The center also computes (for each i) the allocation that would maximize total global utility if i were not in the game; call that sum W_{-i}.
4. Each agent i pays a tax equal to $W_{-i} - B_{-i}$.

In this example, the VCG rule means that each winner would pay a tax equal to the highest reported value among the losers. That is, if I report my value as 5, and that causes someone with value 2 to miss out on an allocation, then I pay a tax of 2. All winners should be happy because they pay a tax that is less than their value, and all losers are as happy as they can be, because they value the goods less than the required tax.

Why is it that this mechanism is truth-revealing? First, consider the payoff to agent i, which is the value of getting an item, minus the tax:

$$v_i(A) - (W_{-i} - B_{-i}) \; . \tag{17.14}$$

Here we distinguish the agent's true utility, v_i, from his reported utility b_i (but we are trying to show that a dominant strategy is $b_i = v_i$). Agent i knows that the center will maximize global utility using the reported values,

$$\sum_j b_j(A) = b_i(A) + \sum_{j \neq i} b_j(A)$$

whereas agent i wants the center to maximize (17.14), which can be rewritten as

$$v_i(A) + \sum_{j \neq i} b_j(A) - W_{-i} \; .$$

Since agent i cannot affect the value of W_{-i} (it depends only on the other agents), the only way i can make the center optimize what i wants is to report the true utility, $b_i = v_i$.

17.7 SUMMARY

This chapter shows how to use knowledge about the world to make decisions even when the outcomes of an action are uncertain and the rewards for acting might not be reaped until many actions have passed. The main points are as follows:

- Sequential decision problems in uncertain environments, also called **Markov decision processes**, or MDPs, are defined by a **transition model** specifying the probabilistic outcomes of actions and a **reward function** specifying the reward in each state.
- The utility of a state sequence is the sum of all the rewards over the sequence, possibly discounted over time. The solution of an MDP is a **policy** that associates a decision with every state that the agent might reach. An optimal policy maximizes the utility of the state sequences encountered when it is executed.
- The utility of a state is the expected utility of the state sequences encountered when an optimal policy is executed, starting in that state. The **value iteration** algorithm for solving MDPs works by iteratively solving the equations relating the utility of each state to those of its neighbors.
- **Policy iteration** alternates between calculating the utilities of states under the current policy and improving the current policy with respect to the current utilities.
- Partially observable MDPs, or POMDPs, are much more difficult to solve than are MDPs. They can be solved by conversion to an MDP in the continuous space of belief

states; both value iteration and policy iteration algorithms have been devised. Optimal behavior in POMDPs includes information gathering to reduce uncertainty and therefore make better decisions in the future.

- A decision-theoretic agent can be constructed for POMDP environments. The agent uses a **dynamic decision network** to represent the transition and sensor models, to update its belief state, and to project forward possible action sequences.
- **Game theory** describes rational behavior for agents in situations in which multiple agents interact simultaneously. Solutions of games are **Nash equilibria**—strategy profiles in which no agent has an incentive to deviate from the specified strategy.
- **Mechanism design** can be used to set the rules by which agents will interact, in order to maximize some global utility through the operation of individually rational agents. Sometimes, mechanisms exist that achieve this goal without requiring each agent to consider the choices made by other agents.

We shall return to the world of MDPs and POMDP in Chapter 21, when we study **reinforcement learning** methods that allow an agent to improve its behavior from experience in sequential, uncertain environments.

BIBLIOGRAPHICAL AND HISTORICAL NOTES

Richard Bellman developed the ideas underlying the modern approach to sequential decision problems while working at the RAND Corporation beginning in 1949. According to his autobiography (Bellman, 1984), he coined the exciting term "dynamic programming" to hide from a research-phobic Secretary of Defense, Charles Wilson, the fact that his group was doing mathematics. (This cannot be strictly true, because his first paper using the term (Bellman, 1952) appeared before Wilson became Secretary of Defense in 1953.) Bellman's book, *Dynamic Programming* (1957), gave the new field a solid foundation and introduced the basic algorithmic approaches. Ron Howard's Ph.D. thesis (1960) introduced policy iteration and the idea of average reward for solving infinite-horizon problems. Several additional results were introduced by Bellman and Dreyfus (1962). Modified policy iteration is due to van Nunen (1976) and Puterman and Shin (1978). Asynchronous policy iteration was analyzed by Williams and Baird (1993), who also proved the policy loss bound in Equation (17.9). The analysis of discounting in terms of stationary preferences is due to Koopmans (1972). The texts by Bertsekas (1987), Puterman (1994), and Bertsekas and Tsitsiklis (1996) provide a rigorous introduction to sequential decision problems. Papadimitriou and Tsitsiklis (1987) describe results on the computational complexity of MDPs.

Seminal work by Sutton (1988) and Watkins (1989) on reinforcement learning methods for solving MDPs played a significant role in introducing MDPs into the AI community, as did the later survey by Barto *et al.* (1995). (Earlier work by Werbos (1977) contained many similar ideas, but was not taken up to the same extent.) The connection between MDPs and AI planning problems was made first by Sven Koenig (1991), who showed how probabilistic STRIPS operators provide a compact representation for transition models (see also Wellman,

1990b). Work by Dean *et al.* (1993) and Tash and Russell (1994) attempted to overcome the combinatorics of large state spaces by using a limited search horizon and abstract states. Heuristics based on the value of information can be used to select areas of the state space where a local expansion of the horizon will yield a significant improvement in decision quality. Agents using this approach can tailor their effort to handle time pressure and generate some interesting behaviors such as using familiar "beaten paths" to find their way around the state space quickly without having to recompute optimal decisions at each point.

As one might expect, AI researchers have pushed MDPs in the direction of more expressive representations that can accommodate much larger problems than the traditional atomic representations based on transition matrices. The use of a dynamic Bayesian network to represent transition models was an obvious idea, but work on **factored MDPs** (Boutilier *et al.*, 2000; Koller and Parr, 2000; Guestrin *et al.*, 2003b) extends the idea to structured representations of the value function with provable improvements in complexity. **Relational MDPs** (Boutilier *et al.*, 2001; Guestrin *et al.*, 2003a) go one step further, using structured representations to handle domains with many related objects.

FACTORED MDP

RELATIONAL MDP

The observation that a partially observable MDP can be transformed into a regular MDP over belief states is due to Astrom (1965) and Aoki (1965). The first complete algorithm for the exact solution of POMDPs—essentially the value iteration algorithm presented in this chapter—was proposed by Edward Sondik (1971) in his Ph.D. thesis. (A later journal paper by Smallwood and Sondik (1973) contains some errors, but is more accessible.) Lovejoy (1991) surveyed the first twenty-five years of POMDP research, reaching somewhat pessimistic conclusions about the feasibility of solving large problems. The first significant contribution within AI was the Witness algorithm (Cassandra *et al.*, 1994; Kaelbling *et al.*, 1998), an improved version of POMDP value iteration. Other algorithms soon followed, including an approach due to Hansen (1998) that constructs a policy incrementally in the form of a finite-state automaton. In this policy representation, the belief state corresponds directly to a particular state in the automaton. More recent work in AI has focused on **point-based** value iteration methods that, at each iteration, generate conditional plans and α-vectors for a finite set of belief states rather than for the entire belief space. Lovejoy (1991) proposed such an algorithm for a fixed grid of points, an approach taken also by Bonet (2002). An influential paper by Pineau *et al.* (2003) suggested generating reachable points by simulating trajectories in a somewhat greedy fashion; Spaan and Vlassis (2005) observe that one need generate plans for only a small, randomly selected subset of points to improve on the plans from the previous iteration for all points in the set. Current point-based methods—such as point-based policy iteration (Ji *et al.*, 2007)—can generate near-optimal solutions for POMDPs with thousands of states. Because POMDPs are PSPACE-hard (Papadimitriou and Tsitsiklis, 1987), further progress may require taking advantage of various kinds of structure within a factored representation.

The online approach—using look-ahead search to select an action for the current belief state—was first examined by Satia and Lave (1973). The use of sampling at chance nodes was explored analytically by Kearns *et al.* (2000) and Ng and Jordan (2000). The basic ideas for an agent architecture using dynamic decision networks were proposed by Dean and Kanazawa (1989a). The book *Planning and Control* by Dean and Wellman (1991) goes

into much greater depth, making connections between DBN/DDN models and the classical control literature on filtering. Tatman and Shachter (1990) showed how to apply dynamic programming algorithms to DDN models. Russell (1998) explains various ways in which such agents can be scaled up and identifies a number of open research issues.

The roots of game theory can be traced back to proposals made in the 17th century by Christiaan Huygens and Gottfried Leibniz to study competitive and cooperative human interactions scientifically and mathematically. Throughout the 19th century, several leading economists created simple mathematical examples to analyze particular examples of competitive situations. The first formal results in game theory are due to Zermelo (1913) (who had, the year before, suggested a form of minimax search for games, albeit an incorrect one). Emile Borel (1921) introduced the notion of a mixed strategy. John von Neumann (1928) proved that every two-person, zero-sum game has a maximin equilibrium in mixed strategies and a well-defined value. Von Neumann's collaboration with the economist Oskar Morgenstern led to the publication in 1944 of the *Theory of Games and Economic Behavior*, the defining book for game theory. Publication of the book was delayed by the wartime paper shortage until a member of the Rockefeller family personally subsidized its publication.

In 1950, at the age of 21, John Nash published his ideas concerning equilibria in general (non-zero-sum) games. His definition of an equilibrium solution, although originating in the work of Cournot (1838), became known as Nash equilibrium. After a long delay because of the schizophrenia he suffered from 1959 onward, Nash was awarded the Nobel Memorial Prize in Economics (along with Reinhart Selten and John Harsanyi) in 1994. The Bayes–Nash equilibrium is described by Harsanyi (1967) and discussed by Kadane and Larkey (1982). Some issues in the use of game theory for agent control are covered by Binmore (1982).

The prisoner's dilemma was invented as a classroom exercise by Albert W. Tucker in 1950 (based on an example by Merrill Flood and Melvin Dresher) and is covered extensively by Axelrod (1985) and Poundstone (1993). Repeated games were introduced by Luce and Raiffa (1957), and games of partial information in extensive form by Kuhn (1953). The first practical algorithm for sequential, partial-information games was developed within AI by Koller *et al.* (1996); the paper by Koller and Pfeffer (1997) provides a readable introduction to the field and describe a working system for representing and solving sequential games.

The use of abstraction to reduce a game tree to a size that can be solved with Koller's technique is discussed by Billings *et al.* (2003). Bowling *et al.* (2008) show how to use importance sampling to get a better estimate of the value of a strategy. Waugh *et al.* (2009) show that the abstraction approach is vulnerable to making systematic errors in approximating the equilibrium solution, meaning that the whole approach is on shaky ground: it works for some games but not others. Korb *et al.* (1999) experiment with an opponent model in the form of a Bayesian network. It plays five-card stud about as well as experienced humans. (Zinkevich *et al.*, 2008) show how an approach that minimizes regret can find approximate equilibria for abstractions with 10^{12} states, 100 times more than previous methods.

Game theory and MDPs are combined in the theory of Markov games, also called stochastic games (Littman, 1994; Hu and Wellman, 1998). Shapley (1953) actually described the value iteration algorithm independently of Bellman, but his results were not widely appreciated, perhaps because they were presented in the context of Markov games. Evolu-

tionary game theory (Smith, 1982; Weibull, 1995) looks at strategy drift over time: if your opponent's strategy is changing, how should you react? Textbooks on game theory from an economics point of view include those by Myerson (1991), Fudenberg and Tirole (1991), Osborne (2004), and Osborne and Rubinstein (1994); Mailath and Samuelson (2006) concentrate on repeated games. From an AI perspective we have Nisan *et al.* (2007), Leyton-Brown and Shoham (2008), and Shoham and Leyton-Brown (2009).

The 2007 Nobel Memorial Prize in Economics went to Hurwicz, Maskin, and Myerson "for having laid the foundations of mechanism design theory" (Hurwicz, 1973). The tragedy of the commons, a motivating problem for the field, was presented by Hardin (1968). The revelation principle is due to Myerson (1986), and the revenue equivalence theorem was developed independently by Myerson (1981) and Riley and Samuelson (1981). Two economists, Milgrom (1997) and Klemperer (2002), write about the multibillion-dollar spectrum auctions they were involved in.

Mechanism design is used in multiagent planning (Hunsberger and Grosz, 2000; Stone *et al.*, 2009) and scheduling (Rassenti *et al.*, 1982). Varian (1995) gives a brief overview with connections to the computer science literature, and Rosenschein and Zlotkin (1994) present a book-length treatment with applications to distributed AI. Related work on distributed AI also goes under other names, including collective intelligence (Tumer and Wolpert, 2000; Segaran, 2007) and market-based control (Clearwater, 1996). Since 2001 there has been an annual Trading Agents Competition (TAC), in which agents try to make the best profit on a series of auctions (Wellman *et al.*, 2001; Arunachalam and Sadeh, 2005). Papers on computational issues in auctions often appear in the ACM Conferences on Electronic Commerce.

EXERCISES

17.1 For the 4×3 world shown in Figure 17.1, calculate which squares can be reached from (1,1) by the action sequence $[Up, Up, Right, Right, Right]$ and with what probabilities. Explain how this computation is related to the prediction task (see Section 15.2.1) for a hidden Markov model.

17.2 Select a specific member of the set of policies that are optimal for $R(s) > 0$ as shown in Figure 17.2(b), and calculate the fraction of time the agent spends in each state, in the limit, if the policy is executed forever. (*Hint*: Construct the state-to-state transition probability matrix corresponding to the policy and see Exercise 15.2.)

17.3 Suppose that we define the utility of a state sequence to be the *maximum* reward obtained in any state in the sequence. Show that this utility function does not result in stationary preferences between state sequences. Is it still possible to define a utility function on states such that MEU decision making gives optimal behavior?

17.4 Sometimes MDPs are formulated with a reward function $R(s, a)$ that depends on the action taken or with a reward function $R(s, a, s')$ that also depends on the outcome state.

a. Write the Bellman equations for these formulations.

b. Show how an MDP with reward function $R(s, a, s')$ can be transformed into a different MDP with reward function $R(s, a)$, such that optimal policies in the new MDP correspond exactly to optimal policies in the original MDP.

c. Now do the same to convert MDPs with $R(s, a)$ into MDPs with $R(s)$.

17.5 For the environment shown in Figure 17.1, find all the threshold values for $R(s)$ such that the optimal policy changes when the threshold is crossed. You will need a way to calculate the optimal policy and its value for fixed $R(s)$. (*Hint*: Prove that the value of any fixed policy varies linearly with $R(s)$.)

17.6 Equation (17.7) on page 654 states that the Bellman operator is a contraction.

a. Show that, for any functions f and g,

$$|\max_a f(a) - \max_a g(a)| \leq \max_a |f(a) - g(a)| \ .$$

b. Write out an expression for $|(B\,U_i - B\,U'_i)(s)|$ and then apply the result from (a) to complete the proof that the Bellman operator is a contraction.

17.7 This exercise considers two-player MDPs that correspond to zero-sum, turn-taking games like those in Chapter 5. Let the players be A and B, and let $R(s)$ be the reward for player A in state s. (The reward for B is always equal and opposite.)

a. Let $U_A(s)$ be the utility of state s when it is A's turn to move in s, and let $U_B(s)$ be the utility of state s when it is B's turn to move in s. All rewards and utilities are calculated from A's point of view (just as in a minimax game tree). Write down Bellman equations defining $U_A(s)$ and $U_B(s)$.

b. Explain how to do two-player value iteration with these equations, and define a suitable termination criterion.

c. Consider the game described in Figure 5.17 on page 197. Draw the state space (rather than the game tree), showing the moves by A as solid lines and moves by B as dashed lines. Mark each state with $R(s)$. You will find it helpful to arrange the states (s_A, s_B) on a two-dimensional grid, using s_A and s_B as "coordinates."

d. Now apply two-player value iteration to solve this game, and derive the optimal policy.

17.8 Consider the 3×3 world shown in Figure 17.14(a). The transition model is the same as in the 4×3 Figure 17.1: 80% of the time the agent goes in the direction it selects; the rest of the time it moves at right angles to the intended direction.

Implement value iteration for this world for each value of r below. Use discounted rewards with a discount factor of 0.99. Show the policy obtained in each case. Explain intuitively why the value of r leads to each policy.

a. $r = 100$

b. $r = -3$

c. $r = 0$

d. $r = +3$

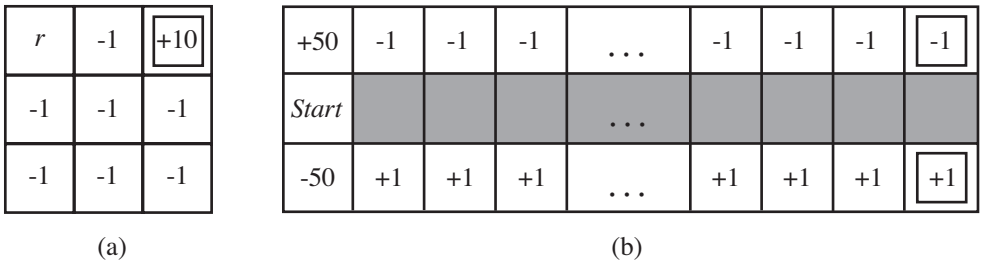


Figure 17.14 (a) 3×3 world for Exercise 17.8. The reward for each state is indicated. The upper right square is a terminal state. (b) 101×3 world for Exercise 17.9 (omitting 93 identical columns in the middle). The start state has reward 0.

17.9 Consider the 101×3 world shown in Figure 17.14(b). In the start state the agent has a choice of two deterministic actions, *Up* or *Down*, but in the other states the agent has one deterministic action, *Right*. Assuming a discounted reward function, for what values of the discount γ should the agent choose *Up* and for which *Down*? Compute the utility of each action as a function of γ. (Note that this simple example actually reflects many real-world situations in which one must weigh the value of an immediate action versus the potential continual long-term consequences, such as choosing to dump pollutants into a lake.)

17.10 Consider an undiscounted MDP having three states, (1, 2, 3), with rewards -1, -2, 0, respectively. State 3 is a terminal state. In states 1 and 2 there are two possible actions: a and b. The transition model is as follows:

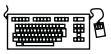
- In state 1, action a moves the agent to state 2 with probability 0.8 and makes the agent stay put with probability 0.2.
- In state 2, action a moves the agent to state 1 with probability 0.8 and makes the agent stay put with probability 0.2.
- In either state 1 or state 2, action b moves the agent to state 3 with probability 0.1 and makes the agent stay put with probability 0.9.

Answer the following questions:

a. What can be determined *qualitatively* about the optimal policy in states 1 and 2?

b. Apply policy iteration, showing each step in full, to determine the optimal policy and the values of states 1 and 2. Assume that the initial policy has action b in both states.

c. What happens to policy iteration if the initial policy has action a in both states? Does discounting help? Does the optimal policy depend on the discount factor?



17.11 Consider the 4×3 world shown in Figure 17.1.

a. Implement an environment simulator for this environment, such that the specific geography of the environment is easily altered. Some code for doing this is already in the online code repository.

b. Create an agent that uses policy iteration, and measure its performance in the environment simulator from various starting states. Perform several experiments from each starting state, and compare the average total reward received per run with the utility of the state, as determined by your algorithm.

c. Experiment with increasing the size of the environment. How does the run time for policy iteration vary with the size of the environment?

17.12 How can the value determination algorithm be used to calculate the expected loss experienced by an agent using a given set of utility estimates U and an estimated model P, compared with an agent using correct values?

17.13 Let the initial belief state b_0 for the 4×3 POMDP on page 658 be the uniform distribution over the nonterminal states, i.e., $\langle \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, 0, 0 \rangle$. Calculate the exact belief state b_1 after the agent moves *Left* and its sensor reports 1 adjacent wall. Also calculate b_2 assuming that the same thing happens again.

17.14 What is the time complexity of d steps of POMDP value iteration for a sensorless environment?

17.15 Consider a version of the two-state POMDP on page 661 in which the sensor is 90% reliable in state 0 but provides no information in state 1 (that is, it reports 0 or 1 with equal probability). Analyze, either qualitatively or quantitatively, the utility function and the optimal policy for this problem.

17.16 Show that a dominant strategy equilibrium is a Nash equilibrium, but not vice versa.

17.17 In the children's game of rock–paper–scissors each player reveals at the same time a choice of rock, paper, or scissors. Paper wraps rock, rock blunts scissors, and scissors cut paper. In the extended version rock–paper–scissors–fire–water, fire beats rock, paper, and scissors; rock, paper, and scissors beat water; and water beats fire. Write out the payoff matrix and find a mixed-strategy solution to this game.

17.18 The following payoff matrix, from Blinder (1983) by way of Bernstein (1996), shows a game between politicians and the Federal Reserve.

	Fed: contract	Fed: do nothing	Fed: expand
Pol: contract	$F = 7, P = 1$	$F = 9, P = 4$	$F = 6, P = 6$
Pol: do nothing	$F = 8, P = 2$	$F = 5, P = 5$	$F = 4, P = 9$
Pol: expand	$F = 3, P = 3$	$F = 2, P = 7$	$F = 1, P = 8$

Politicians can expand or contract fiscal policy, while the Fed can expand or contract monetary policy. (And of course either side can choose to do nothing.) Each side also has preferences for who should do what—neither side wants to look like the bad guys. The payoffs shown are simply the rank orderings: 9 for first choice through 1 for last choice. Find the Nash equilibrium of the game in pure strategies. Is this a Pareto-optimal solution? You might wish to analyze the policies of recent administrations in this light.

17.19 A Dutch auction is similar in an English auction, but rather than starting the bidding at a low price and increasing, in a Dutch auction the seller starts at a high price and gradually lowers the price until some buyer is willing to accept that price. (If multiple bidders accept the price, one is arbitrarily chosen as the winner.) More formally, the seller begins with a price p and gradually lowers p by increments of d until at least one buyer accepts the price. Assuming all bidders act rationally, is it true that for arbitrarily small d, a Dutch auction will always result in the bidder with the highest value for the item obtaining the item? If so, show mathematically why. If not, explain how it may be possible for the bidder with highest value for the item not to obtain it.

17.20 Imagine an auction mechanism that is just like an ascending-bid auction, except that at the end, the winning bidder, the one who bid b_{max}, pays only $b_{max}/2$ rather than b_{max}. Assuming all agents are rational, what is the expected revenue to the auctioneer for this mechanism, compared with a standard ascending-bid auction?

17.21 Teams in the National Hockey League historically received 2 points for winning a game and 0 for losing. If the game is tied, an overtime period is played; if nobody wins in overtime, the game is a tie and each team gets 1 point. But league officials felt that teams were playing too conservatively in overtime (to avoid a loss), and it would be more exciting if overtime produced a winner. So in 1999 the officials experimented in mechanism design: the rules were changed, giving a team that loses in overtime 1 point, not 0. It is still 2 points for a win and 1 for a tie.

a. Was hockey a zero-sum game before the rule change? After?

b. Suppose that at a certain time t in a game, the home team has probability p of winning in regulation time, probability $0.78 - p$ of losing, and probability 0.22 of going into overtime, where they have probability q of winning, $.9 - q$ of losing, and .1 of tying. Give equations for the expected value for the home and visiting teams.

c. Imagine that it were legal and ethical for the two teams to enter into a pact where they agree that they will skate to a tie in regulation time, and then both try in earnest to win in overtime. Under what conditions, in terms of p and q, would it be rational for both teams to agree to this pact?

d. Longley and Sankaran (2005) report that since the rule change, the percentage of games with a winner in overtime went up 18.2%, as desired, but the percentage of overtime games also went up 3.6%. What does that suggest about possible collusion or conservative play after the rule change?