10

Solution Methods for Decision Graphs

In Chapter 9 we presented graphical languages for modeling decision problems. The languages ease the burden of specifying the problem and transfer the complexity of the problem to the computer. For problems with a finite time horizon, the computer may fold out the specification to a decision tree and determine an optimal strategy by averaging out and folding back as described in Section 9.3.3. However, the calculations may be intractable, and in this chapter we present alternative methods exploiting symmetries in the decision problem. Sections 10.1–10.3 are devoted to solution methods for influence diagrams. Section 10.4 presents a method for solving unconstrained influence diagrams. In Section 10.5 we consider decision theoretic troubleshooting, which has next to no temporal ordering, and for which the decision trees tend to be intractably large. In Section 10.6 we present two methods for solving MDPs, and a method for solving POMDPs is indicated. The last section presents LIMIDs, which is a way of approximating influence diagrams by limiting the memory of the decision maker.

10.1 Solutions to Influence Diagrams

An influence diagram has three types of nodes: *chance nodes*, *decision nodes*, and *utility nodes*. The set of chance nodes is denoted by $\mathcal{U}_C$, the set of decision nodes is denoted by $\mathcal{U}_D$ and the set of utility nodes is denoted by $\mathcal{U}_V$. The *universe* is $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$. We shall also refer to the members of $\mathcal{U}$ as the variables of the influence diagram.

The decision nodes have a temporal order, $D_1, \ldots, D_n$, and the chance nodes are partitioned according to when they are observed: $\mathcal{I}_0$ is the set of chance nodes observed prior to any decision, $\ldots, \mathcal{I}_i$ is the set of chance nodes observed after D_i is taken and before the decision D_{i+1} is taken. Finally, $\mathcal{I}_n$ is the set of chance nodes never observed or observed after the last decision. That is, we have a partial temporal ordering $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \ldots \prec D_n \prec \mathcal{I}_n$.

Recall that an influence diagram is constructed so that if $A \prec D_i$, then there is a directed path from A to D_i.

We shall in this chapter use the influence diagram DI in Figure 10.1 as a standard example, where $\mathcal{I}_0 = \emptyset$, $\mathcal{I}_1 = \{T\}$, and $\mathcal{I}_2 = \{A, B, C\}$. In order not to make things unnecessarily complicated, all variables in DI are binary.

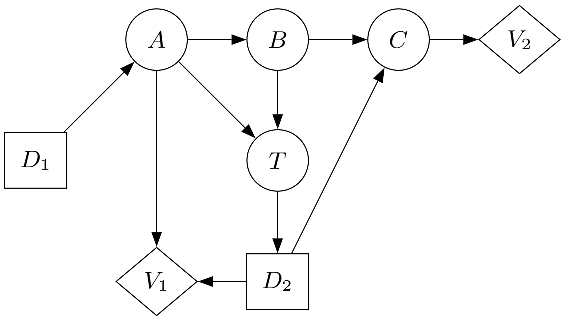

Fig. 10.1. The example influence diagram, DI.

As in Bayesian networks, the graphical representation of influence diagrams supports an analysis of conditional independence. However, d-separation for influence diagrams is performed slightly differently from the way it is done for Bayesian networks: ignore the utility nodes, and since the links into decision nodes encode only information precedence, they shall also be ignored.

For the DI example, we can perform d-separation analysis on Figure 10.1. We get, for example, that C is d-separated from T given B (note that you need not condition on D_2, since the link from T to D_2 is ignored). Also, A and T are d-separated from D_2. This means that if I *perform an action* from D_2, then this action has no impact on T. Note that this is different from: if I am told what action from D_2 was performed, what can I infer about T? If, for example, I know that the decision maker maximizes expected utilities, I may be able to infer a great deal about T.

Decision variables play a different role from that played by chance variables. For chance variables you ask the question, may information about node A change my belief about node B? For decision variables the question is, may an action from D have consequences for node B? Although the two concepts are different, they are in the case of influence diagrams not in conflict. In general, effects of decisions cannot "go back in time":

Proposition 10.1. *Let $A \in \mathcal{I}_i$ and let D_j be a decision variable with $i < j$. Then*

(i) A and D_j are d-separated and hence

$$P(A \mid D_j) = P(A).$$

(ii) Let W be any set of variables prior to D_j in the temporal ordering. Then A and D_j are d-separated given W and hence

$$P(A \mid D_j, W) = P(A \mid W).$$

Proof.

(i) Since D_j has no parents, any impact from D_j must follow the direction of a link from D_j. The only way the impact can start going in the opposite direction from that of a link is if it meets a converging connection at a chance variable B, and then it can do so only if either B or one of its children C has received evidence. Since D_j is the only variable we condition on, this cannot happen. Hence if D_j and A are not d-separated, there must be a directed path from D_j to A. Since $A \prec D_j$ in the temporal ordering, there is a directed path from A to D_j, and since the graph is acyclic, there cannot be a directed path from D_j to A.

(ii) We argue in the same way as for *(i)*. By following directions of links from D_j, we can only start going opposite to the direction by meeting evidence. Since all evidence is prior in the temporal ordering, we know from *(i)* that we cannot meet it.

$\square$

10.1.1 The Chain Rule for Influence Diagrams

For Bayesian networks we have that $P(\mathcal{U})$ is the product of all probability potentials attached to the variables in the network. For influence diagrams we have a similar theorem. Again, decision variables act differently from chance variables. since a decision variable eventually will come under my control, it requires no prior probabilities. Also, it has no meaning to attach a probability distribution to a chance variable A effected by a decision variable D, unless a decision has been taken and the action performed. So in Figure 10.1 it has no meaning to consider $P(A)$ or $P(A, D)$. What is meaningful is $P(A \mid d)$ for all $d \in D$, and we may lump the probabilities for all decisions of D together in the expression $P(A \mid D)$.

Theorem 10.1 (The chain rule for influence diagrams). *Let ID be an influence diagram with universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$. Then*

$$P(\mathcal{U}_C \mid \mathcal{U}_D) = \prod_{X \in \mathcal{U}_C} P(X \mid \mathrm{pa}(X)).$$

Proof. Let us first look at the influence diagram DI. From the fundamental rule we have

$$\begin{aligned}P(C,T,B,A \mid D_1,D_2) &= P(C \mid T,B,A,D_1,D_2)P(T,B,A \mid D_1,D_2)\\ &= P(C \mid T,B,A,D_1,D_2)P(T \mid B,A,D_1,D_2)\\ &\quad \times P(B \mid A,D_1,D_2)P(A \mid D_1,D_2). \qquad (10.1)\end{aligned}$$

Since C is d-separated from A,T, and D_1 given B and D_2, we have

$$P(C \mid T,B,A,D_1,D_2) = P(C \mid B,D_2).$$

We also have

$$\begin{aligned}P(T \mid B,A,D_1,D_2) &= P(T \mid B,A),\\ P(B \mid A,D_1,D_2) &= P(B \mid A),\\ P(A \mid D_1,D_2) &= P(A \mid D_1).\end{aligned}$$

Substituting in equation (10.1) yields

$$P(C,B,T,A \mid D_1,D_2) = P(C \mid B,D_2)P(T \mid B,A)P(B \mid A)P(A \mid D_1,),$$

which is the product of the probability potentials for DI.

A general proof can follow another line of reasoning. Let $\mathbf{d}$ be a particular configuration of decisions. By inserting them in the influence diagram ID, you get a Bayesian network representing $P(\mathcal{U}_C \mid \mathbf{d})$, the joint probability of $\mathcal{U}_C$, under the condition that the decisions $\mathbf{d}$ are taken. Using the chain rule for Bayesian networks, you infer that $P(\mathcal{U}_C \mid \mathbf{d})$ is the product of all probability potentials attached to the decision variables instantiated to $\mathbf{d}$. Since this holds for all instantiations of $\mathcal{U}_\mathcal{D}$, you get the result. □

10.1.2 Strategies and Expected Utilities

To solve an influence diagram, you may unfold it into a decision tree and solve it. In Figure 10.2 we have unfolded DI from Figure 10.1.

When solving the decision tree in Figure 10.2, we start at the leaves and work toward the root (see Section 9.3.3). Consider the path (d_1^1, t_1). We wish to compute the expected utility of performing action d_1^2 given (d_1^1, t_1). We have

$$\mathrm{EU}(d_1^2 \mid d_1^1, t_1) = \sum_{A,C} P(A,C \mid d_1^1, t_1, d_1^2)(V_1(A, d_1^2) + V_2(C)).$$

For the action d_2^2, we have

$$\mathrm{EU}(d_2^2 \mid d_1^1, t_1) = \sum_{A,C} P(A,C \mid d_1^1, t_1, d_2^2)(V_1(A, d_2^2) + V_2(C)).$$

Taken together, we write

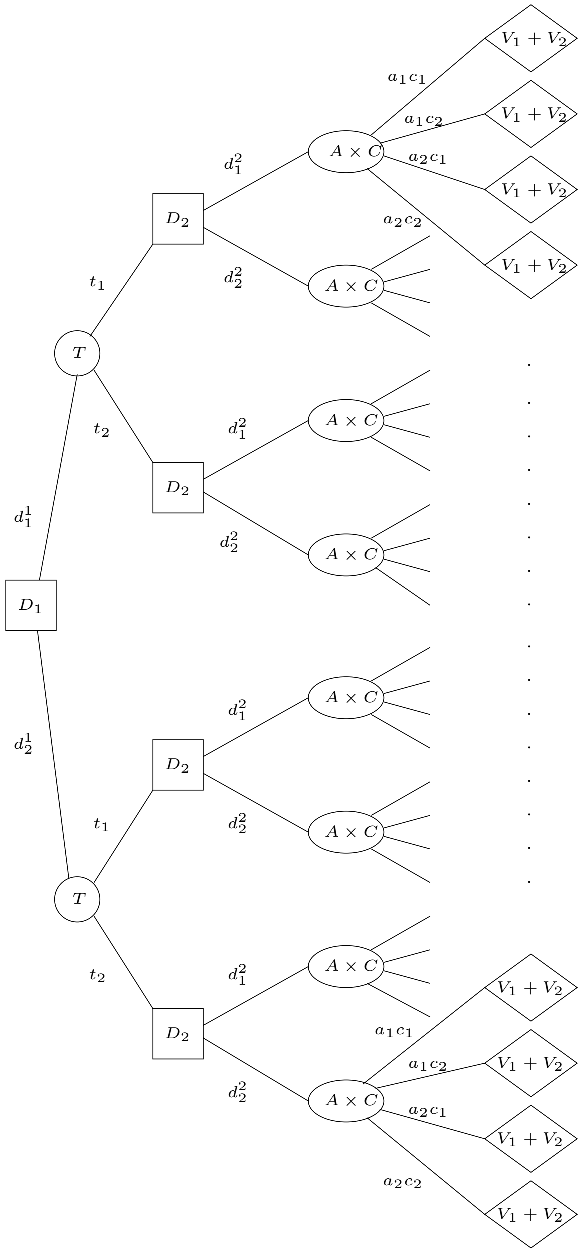

Fig. 10.2. *DI* from Figure 10.1 unfolded into a decision tree. Note that to reduce the size of the decision tree the last chance node in each path is defined as the Cartesian product of A and C, and that the utilities in the leaves are the sums of V_1 and V_2.

$$\mathrm{EU}(D_2 \mid d_1^1, t_1) = \sum_{A,C} P(A, C \mid d_1^1, t_1, D_2)(V_1(A, D_2) + V_2(C)).$$

We choose the action of maximal expected utility, and we get a decision rule for D_2 with $D_1 = d_1^1$ and $T = t_1$

$$\delta_2(d_1^1, t_1) = \arg\max_{D_2} \mathrm{EU}(D_2 \mid d_1^1, t_1).$$

If there are several decisions yielding the maximum, either of them will do. The maximal expected utility from D_2 given (d_1^1, t_1) is

$$\rho_2(d_1^1, t_1) = \max_{D_2} \sum_{A,C} P(A, C \mid d_1^1, t_1, D_2)(V_1(A, D_2) + V_2(C)).$$

Generalizing these two formulas to any path over D_1, T, we get a policy for D_2

$$\begin{aligned} \delta_2(D_1, T) &= \arg\max_{D_2} \mathrm{EU}(D_2 \mid D_1, T) \\ &= \arg\max_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)), \end{aligned}$$

and a new utility function

$$\rho_2(D_1, T) = \max_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)), \quad (10.2)$$

which gives the expected utilities when we know the values of (D_1, T). The decision tree in Figure 10.2 can now be reduced to the one in Figure 10.3.

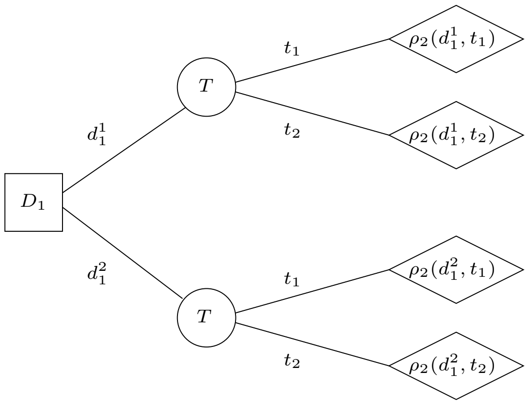

Fig. 10.3. The decision tree from Figure 10.2 with D_2 replaced by a utility function reflecting that the policy δ_2 for D_2 is followed.

Next, look at the decision D_1 as in Figure 10.3. If we take the action d_1^1, we get the expected utility

$$\mathrm{EU}(d_1^1) = P(t_1 \,|\, d_1^1)\rho_2(d_1^1, t_1) + P(t_2 \,|\, d_1^1)\rho_2(d_1^1, t_2),$$

which can also be written

$$\mathrm{EU}(D_1) = \sum_T P(T \,|\, D_1)\rho_2(D_1, T).$$

The policy for D_1 is

$$\delta_1 = \arg\max_{D_1} \sum_T P(T \,|\, D_1)\rho_2(D_1, T),$$

and the expected utility of performing optimal decisions is

$$\rho_1 = \max_{D_1} \sum_T P(T \,|\, D_1)\rho_2(D_1, T). \tag{10.3}$$

So far we have written various expressions without really connecting them to the potentials from the influence diagram. In principle, all probabilities in the expressions can be calculated from the influence diagram by inserting and propagating evidence. However, by taking a closer look at equation (10.3) we can make a much tighter connection between the specification of the influence diagram and its solution: By combining equation (10.2) and equation (10.3), we get

$$\begin{aligned}
\rho_1 &= \max_{D1} \sum_T P(T \,|\, D_1) \max_{D_2} \sum_{A,C} P(A, C \,|\, D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \sum_{A,C} P(T \,|\, D_1)P(A, C \,|\, D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \sum_{A,C} P(T \,|\, D_1, D_2)P(A, C \,|\, D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \sum_{A,C} P(A, C, T \,|\, D_1, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \sum_{A,B,C} P(A, B, C, T \,|\, D_1, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \sum_{A,B,C} P(\mathcal{U}_C \,|\, \mathcal{U}_D)(V_1(A, D_2) + V_2(C)).
\end{aligned}$$

The formula for δ_1 is

$$\delta_1 = \arg\max_{D_1} \sum_T \max_{D_2} \sum_{A,B,C} P(\mathcal{U}_C \,|\, \mathcal{U}_D)(V_1(A, D_2) + V_2(C)).$$

For the policy δ_2 we have

$$\delta_2(D_1, T) = \arg\max_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)).$$

We can multiply inside "$\arg\max_{D_2}$" with anything not varying with D_2:

$$\begin{aligned}\delta_2(D_1, T) &= \arg\max_{D_2} P(T \mid D_1) \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\ &= \arg\max_{D_2} \sum_{A,C} P(T \mid D_1, D_2) P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\ &= \arg\max_{D_2} \sum_{A,C} P(A, T, C \mid D_1, D_2)(V_1(A, D_2) + V_2(C)) \\ &= \arg\max_{D_2} \sum_{A,B,C} P(\mathcal{U}_C \mid \mathcal{U}_D)(V_1(A, D_2) + V_2(C)),\end{aligned}$$

and the similarity with the formula for δ_1 is transparent. Similar calculations yield for ρ_2,

$$\rho_2(D_1, T) = \frac{1}{P(T \mid D_1)} \max_{D_2} \sum_{A,B,C} P(\mathcal{U}_C \mid \mathcal{U}_D)(V_1(A, D_2) + V_2(C)).$$

Theorem 10.2. *Let ID be an influence diagram over $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$ and $\mathcal{U}_V = \{V_i\}$. Let the temporal order of the variables be described as $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$ and let $V = \sum_i V_i$. Then:*

(i) An optimal policy for D_i is

$$\delta_i(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1}) = \arg\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \mid \mathcal{U}_D) V.$$

(ii) The expected utility from following the policy δ_i (and acting optimally in the future) is

$$\rho_i(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1}) = \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \mid D_1, \ldots, D_{i-1})} \max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \mid \mathcal{U}_D) V,$$

and the strategy for ID consisting of an optimal policy for each decision yields the maximum expected utility:

$$\mathrm{MEU}(ID) = \sum_{\mathcal{I}_0} \max_{D_1} \sum_{\mathcal{I}_1} \max_{D_2} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \mid \mathcal{U}_D) V.$$

Proof. We start with the last decision D_n. We have for the expected utility given the past

$$\begin{aligned}
&\mathrm{EU}(D_n \,|\, \mathcal{I}_0, D_1, \ldots, D_{n-1}, \mathcal{I}_{n-1}) \\
&\qquad = \sum_{\mathcal{I}_n} P(\mathcal{I}_n \,|\, \mathcal{I}_0, D_1, \ldots, D_{n-1}, \mathcal{I}_{n-1}, D_n)V \\
&\qquad = \sum_{\mathcal{I}_n} \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_n)} P(\mathcal{I}_n, \mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_n)V \\
&\qquad = \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_{n-1})} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V.
\end{aligned}$$

In the last expression we used that $P(\mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_n) = P(\mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_{n-1})$. We now get

$$\begin{aligned}
&\rho_n(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{n-1}) \\
&\qquad = \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \,|\, D_1, \ldots, D_{n-1})} \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V,
\end{aligned}$$

and

$$\begin{aligned}
&\delta_n(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{n-1}) \\
&\qquad = \arg\max_{D_n} \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{n-1} \,|\, D_1, \ldots, D_{n-1})} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V \\
&\qquad = \arg\max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V.
\end{aligned}$$

Next, assume the theorem to hold for $i+1, \ldots, n$ and consider decision D_i. We have

$$\begin{aligned}
&\mathrm{EU}(D_i \,|\, \mathcal{I}_0, D_1, \ldots, D_{n-1}, \mathcal{I}_{i-1}) \\
&\qquad = \sum_{\mathcal{I}_i} P(\mathcal{I}_i \,|\, \mathcal{I}_0, D_1, \ldots, D_{i-1}, \mathcal{I}_{i-1}, D_i)\rho_{i+1}(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_i) \\
&\qquad = \sum_{\mathcal{I}_i} \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \,|\, D_1, \ldots, D_i)} P(\mathcal{I}_i, \mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \,|\, D_1, \ldots, D_i) \\
&\qquad\quad \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_i \,|\, D_1, \ldots, D_i)} \max_{D_{i+1}} \sum_{\mathcal{I}_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V \\
&\qquad = \sum_{\mathcal{I}_i} \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \,|\, D_1, \ldots, D_i)} \max_{D_{i+1}} \sum_{\mathcal{I}_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V \\
&\qquad = \frac{1}{P(\mathcal{I}_0, \ldots, \mathcal{I}_{i-1} \,|\, D_1, \ldots, D_{i-1})} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \sum_{\mathcal{I}_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V,
\end{aligned}$$

and we get the formulas in *(i)* and *(ii)*.

Since we have repeatedly determined a policy maximizing the expected utility regardless of the past, no other set of policies can give a higher expected utility. The formula for $\mathrm{MEU}(ID)$ is the formula from *(ii)* for ρ_0. It is calculated by taking $\rho_1(D_1)$, multiplying by $P(\mathcal{I}_0)$, and marginalizing $\mathcal{I}_0$ out:

$$\begin{aligned}\mathrm{MEU}(ID) &= \sum_{\mathcal{I}_0} P(\mathcal{I}_0)\rho_1(\mathcal{I}_0)\\ &= \sum_{\mathcal{I}_o} P(\mathcal{I}_0)\frac{1}{P(\mathcal{I}_0)} \max_{D_1}\sum_{\mathcal{I}_1}\max_{D_2}\cdots\max_{D_n}\sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V\\ &= \sum_{\mathcal{I}_0}\max_{D_1}\sum_{\mathcal{I}_1}\max_{D_2}\cdots\max_{D_n}\sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D)V.\end{aligned}$$

□

Since $P(\mathcal{U}_C \,|\, \mathcal{U}_D)$ is the product of all probability distributions attached to ID, we have a method for calculating ρ_i as well as δ_i. The method specifies that you may start with the product of all probability potentials and then marginalize out in reverse temporal order where chance variables are sum-marginalized and decision variables are max-marginalized. Each time an $\mathcal{I}_i$ is marginalized out, the result is used to determine a policy for D_i.

The method has the same problem as the method for Bayesian networks, namely that $P(\mathcal{U}_C \,|\, \mathcal{U}_D)$ may be an intractably large table, and we therefore have to look for methods that reduce the size of the domains to deal with. We shall consider this task in detail in Section 10.2.

10.1.3 An Example

The influence diagram DI in Figure 10.1 has the potentials in Table 10.1. Using Theorem 10.2 we get $\delta_2(D_1, T)$ and $\rho_2(D_1, T)$ as listed in Table 10.2.

$A \setminus D_1$	d_1^1	d_2^1
y	0.2	0.8
n	0.8	0.2

$P(A \,|\, D_1)$

$B \setminus A$	y	n
y	0.8	0.2
n	0.2	0.8

$P(B \,|\, A)$

$A \setminus B$	y	n
y	(0.9, 0.1)	(0.5, 0.5)
n	(0.5, 0.5)	(0.1, 0.9)

$P(T \,|\, A, B)$

$B \setminus D_2$	d_1^2	d_2^2
y	(0.9, 0.1)	(0.5, 0.5)
n	(0.5, 0.5)	(0.9, 0.1)

$P(C \,|\, D_2, B)$

$A \setminus D_2$	d_1^2	d_2^2
y	3	0
n	0	2

$V(A, D_2)$

$V_2(C) = (10, 0)$

Table 10.1. Potentials for DI.

$T \setminus D_1$	d_1^1	d_2^1
y	d_1^2	d_1^2
n	d_2^2	d_2^2

$\delta_2(D_1, T)$

$T \setminus D_1$	d_1^1	d_2^1
y	9.51	11.29
n	10.34	8.97

$\rho_1(D_1, T)$

Table 10.2. $\delta_2(D_1, T)$ and $\rho_2(D_1, T)$ for DI.

Finally, we get $\delta_1 = d_2^1$ and MEU(DI) = 10.58. Note that $\delta_2(D_1, T)$ has the property that the state of T alone determines the decision to choose, hence we can remove D_1 from the domain of δ_2. This phenomenon can sometimes be determined from the d-separation properties of the influence diagram (see Figure 10.4 and Section 11.2), and we say that this part of the past is not *required* for the decision in question. For DI it cannot be deduced from the structure; the potentials happened to cause it.

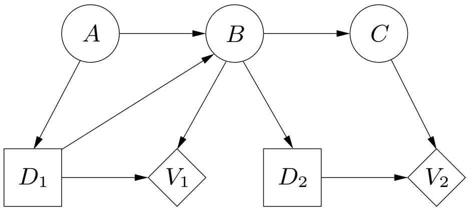

Fig. 10.4. An influence diagram in which D_1 is not required for D_2.

10.2 Variable Elimination

The method for solving influence diagrams has many similarities with the junction tree propagation algorithm for Bayesian networks: you start off with a set of potentials, and you eliminate one variable at a time. There are, however, differences. First of all, the elimination order is constrained by the temporal order. Since max-marginalization and sum-marginalization do not commute, you have to do it in an order whereby you first sum-marginalize $\mathcal{I}_n$, then max-marginalize D_n, sum-marginalize $\mathcal{I}_{i-1}$, etc. This type of elimination order is called a *strong elimination order*. Furthermore, you have two types of potentials to deal with. Also, you need to eliminate in only one direction; this corresponds to CollectEvidence.

We shall first analyze the calculations in eliminating a variable. Let Φ be a set of probability potentials and Ψ a set of utility potentials. The two sets represent the expression $\prod \Phi(\sum \Psi)$, the product of all probability potentials multiplied by the sum of all utility potentials.

Now assume that we shall calculate $\sum_X \prod \Phi(\sum \Psi)$ for some chance variable X. To do that we partition Φ into two sets: Φ_X, which is the set of potentials with X in the domain, and $\Phi^* = \Phi \setminus \Phi_X$. The set Ψ is in the same way divided up in the two sets Ψ_X and Ψ^*. Set $\phi_X = \sum_X \prod \Phi_X$ and $\psi_X = \sum_X \prod \Phi_X(\sum \Psi_X)$. Using the distributive law we get

$$\begin{aligned}\sum_X \prod \Phi \left(\sum \Psi\right) &= \prod \Phi^* \sum_X \left(\prod \Phi_X \left(\sum \Psi^* + \sum \Psi_X\right)\right)\\ &= \prod \Phi^* \left(\left(\sum \Psi^*\right) \sum_X \left(\prod \Phi_X\right) + \sum_X \prod \Phi_X \left(\sum \Psi_X\right)\right)\\ &= \prod \Phi^* \left(\left(\sum \Psi^*\right) \phi_X + \psi_X\right)\\ &= \prod \Phi^* \phi_X \left(\sum \Psi^* + \frac{\psi_X}{\phi_X}\right).\end{aligned}$$

We see that the result of eliminating the chance variable X is that Φ_X is removed from the set of probability potentials and substituted with ϕ_X. For the set of utility potentials, Ψ_X is removed and $\frac{\psi_X}{\phi_X}$ is added.

Let D be a decision variable. We again divide the potentials into Φ_D, Φ^* and Ψ_D, Ψ^*. Since all variables coming after D in the temporal ordering have been eliminated when we are about to eliminate D, it follows that $\prod \Phi_D$ does not vary with D (See Exercise 10.3). So taking $\max_D$ of $\prod \Phi_D$ is an almost empty operation; it only removes D from the domain. Using the distributive law for max, setting $\phi_D = \max_D \prod \Phi_D$ and $\psi_D = \max_D \prod \Phi_D(\sum \Psi_D)$, and exploiting that $\prod \Phi_D(\sum \Psi^*)$ does not vary with D, we get

$$\begin{aligned}\max_D \prod \Phi \left(\sum \Psi\right) &= \prod \Phi^* \max_D \left(\prod \Phi_D \left(\sum \Psi^* + \sum \Psi_D\right)\right)\\ &= \prod \Phi^* \left(\max_D \prod \Phi_D \left(\sum \Psi^*\right) + \max_D \prod \Phi_D \left(\sum \Psi_D\right)\right)\\ &= \prod \Phi^* \left(\phi_D \left(\sum \Psi^*\right) + \psi_D\right)\\ &= \prod \Phi^* \phi_D \left(\sum \Psi^* + \frac{\psi_D}{\phi_D}\right).\end{aligned}$$

The result is similar to the result for sum-elimination. To sum up:

Algorithm 10.1 [Variable elimination for influence diagrams] *You work with two sets of potentials: Φ, the set of probability potentials; Ψ, the set of utility potentials. When a variable X is eliminated, the potential sets are modified in the following way:*

1.

$$\begin{aligned}\Phi_X &:= \{\phi \in \Phi \mid X \in \mathrm{dom}(\phi)\};\\ \psi_X &:= \{\psi \in \Psi \mid X \in \mathrm{dom}(\psi)\}.\end{aligned}$$

2. If X is a chance variable, then

$$\phi_X := \sum_X \prod \Phi_X;$$
$$\psi_X := \sum_X \prod \Phi_X \left(\sum \Psi_X\right).$$

If X is a decision variable, then

$$\phi_X := \max_X \prod \Phi_X;$$
$$\psi_X := \max_X \prod \Phi_X \left(\sum \Psi_X\right).$$

3.

$$\Phi := (\Phi \setminus \Phi_X) \cup \{\phi_X\}$$
$$\Psi := (\Psi \setminus \Psi_X) \cup \left\{\frac{\psi_X}{\phi_X}\right\}.$$

□

The influence diagram is solved by repeatedly eliminating variables according to a strong elimination order.

10.2.1 Strong Junction Trees

The considerations on triangulated graphs and junction trees (see Section 4.4) can be applied when the method above is used for solving influence diagrams. The considerations shall not be repeated here. Consider now the influence diagram in Figure 10.5

When solving the influence diagram, you first establish the moral graph: for each potential you link all variables in the domain. For the graph it means that you remove information links, add a link for each pair of nodes with a common child (including a common utility node), and finally remove the directions and the utility nodes. It is done in Figure 10.6 for the influence diagram in Figure 10.5.

As opposed to Bayesian networks, we cannot choose any elimination order for the triangulation. We have to follow a strong elimination order: first eliminate $\mathcal{I}_n$ (in any order), then eliminate D_n, then $\mathcal{I}_{n-1}$ and so on (if some $\mathcal{I}_i$ is empty, we may permute the elimination of D_{i+1} and D_i). The resulting triangulation is called a *strong triangulation*. Figure 10.7 shows the strong triangulation resulting from eliminating the nodes in the moral graph in Figure 10.5 through the strong elimination order $A, L, I, J, K, H, C, D, D_4, G, D_3, D_2, E, F, D_1, B$.

If you use the method for constructing the join trees from Section 4.3.1, the result of a strong triangulation is called a *strong junction tree* with the

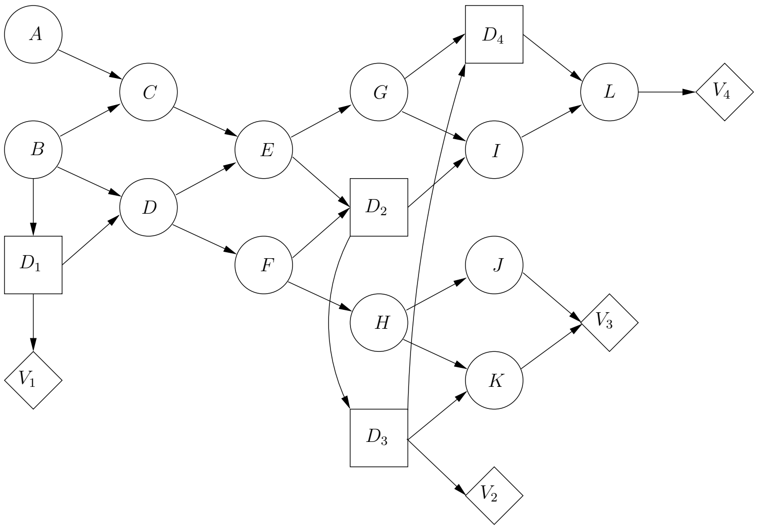

Fig. 10.5. The influence diagram from Figure 9.22.

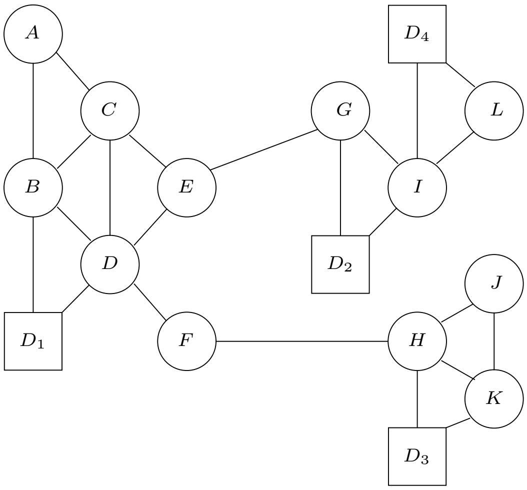

Fig. 10.6. The moral graph for the influence diagram in Figure 10.5.

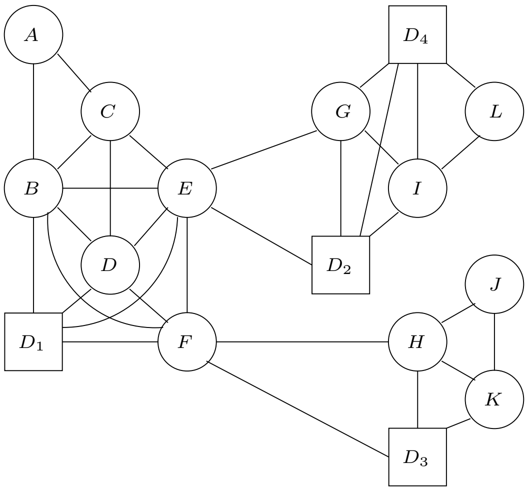

Fig. 10.7. A strong triangulation of the graph in Figure 10.6.

last clique constructed in the strong elimination order serving as a *strong root*. A junction tree with a strong root R has the following property: for any two neighboring cliques C, C' with separator S and C' closest to R, it holds that the variables in S do not appear after the variables in $C \setminus S$ in the temporal order. This property ensures that when COLLECTEVIDENCE(R) is called, then whenever a variable is eliminated the appropriate potentials are present. Figure 10.8 shows a strong junction tree for the graph in Figure 10.7.

Note: Although the influence diagram prescribes a specific order of the decisions, it happens that some decisions are independent such that the order may be altered without changing the strategy or the MEU. This is sometimes detected in constructing a strong junction tree. That is, if you follow the method from Section 4.3.1, you may get a tree in which the decision nodes are eliminated in two different branches (as is the case in Figure 10.8, where the elimination of D_3 can be done independently of D_2 and D_4).

From the strong junction tree, you can construct elimination sequences that do not meet the temporal constraints (the elimination sequence $J, K, H, D_3, A, C, L, I, D_4, G, D_2, D, E, F, D_1, B$ is a perfect elimination sequence ending with B, but it does not follow the temporal order). Since the result of COLLECTEVIDENCE(R) is independent of the actual order of messages sent, all elimination sequences allowed by the strong junction tree give the same result (as long as the elimination order inside each clique obeys the temporal constraints). This means that the strong junction tree in Figure 10.8 discloses that D_3 and D_4 are independent, and the temporal order can be relaxed to a partial ordering of the decision nodes.

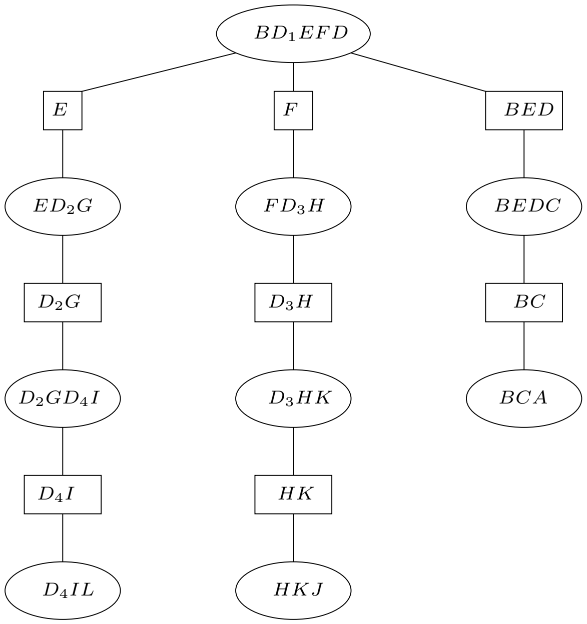

Fig. 10.8. A strong junction tree for the graph in Figure 10.7.

It may also happen that the strong junction tree does not allow for a strong elimination sequence when CollectEvidence(R) is called. An example is given in Figure 10.9, where C and F are the first variables to be eliminated according to the temporal ordering, but in the strong junction tree, C is eliminated after D_4. However, this is not a problem, since C cannot affect the policy for D_4 (see Section 11.2).

10.2.2 Required Past

As noted previously, the domain of a policy for a decision variable D_1 is in general $(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1})$, but a strong elimination order can reveal reductions of the domain: whenever D_i is eliminated, you consider only the potentials with D_i in the domain. The required past must therefore be a subset of the union of these domains, and thus part of the clique closest to the strong root containing D_i.

With the strong elimination ordering $A, L, I, J, K, H, C, D, D_4, G, D_3, D_2, E, F, D_1, B$ for the influence diagram in Figure 10.5, we get the following policies for the decision variables: $\delta_4(G, D_2)$, $\delta_3(F)$, $\delta_2(E)$, and $\delta_1(B)$. Here we see that the policy for D_4, say, contains only two variables as opposed to the seven variables that constitute the past for D_4.

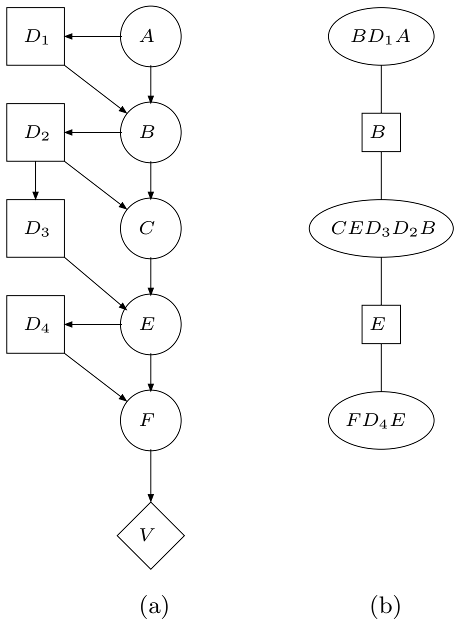

Fig. 10.9. An influence diagram (a) with a strong junction tree (b) for which CollectEvidence(R) does not initiate a strong elimination sequence meeting the temporal constraints: C should be eliminated before D_4.

This analysis does not guarantee minimal policy domains, as can be seen from the influence diagram in Figure 10.10. We shall return to this issue in Section 11.2.

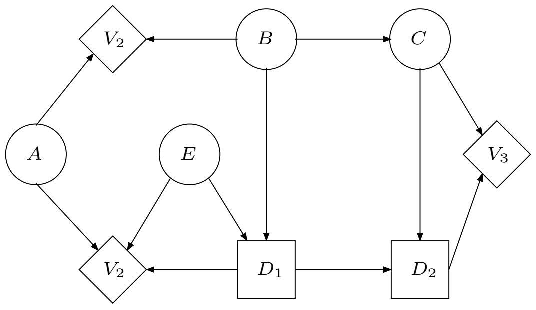

Fig. 10.10. The minimal domain of the policy for D_1 contains only the variable E, but a strong elimination ordering would produce a policy over E and B.

10.2.3 Policy Networks

When a strategy for an influence diagram has been determined, we have a policy δ_i for each decision node D_i. The domain of δ_i is $(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1})$, but as shown above (and explained in Section 11.2) we may be able to reduce it so that it contains only the required variables, denoted by $\mathrm{req}(D_i)$.

A decision variable can together with its policy be represented in a Bayesian network.

Definition 10.1. *Let D be a decision variable with policy δ_D. The* chance-variable representation of D *is the result of the following construction: Substitute D with a chance variable D^* having parents* $\mathrm{req}(D)$*, and assign D^* the conditional probability distribution $P(D^* \mid \mathrm{req}(D))$:*

$$P(d|\mathbf{r}) = \begin{cases} 1 & \textit{if } \delta_D(\mathbf{r}) = d, \\ 0 & \textit{otherwise.} \end{cases}$$

If all decision variables are substituted with their chance-variable representations, we obtain a so-called policy network for the influence diagram.

Definition 10.2. *Let I be an influence diagram over $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$. A* policy network *for ID (denoted by I^*) is a Bayesian network over $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D^*$ in which all decision variables D_i have been substituted with their chance-variable representations. The probability potentials from I are kept (with D_js replaced by D_j^*).*

Figure 10.11 shows the policy network for the influence diagram in Figure 10.5 with the policy domains determined in Section 10.2.2.

Example 10.1. A farmer has a wheat field. Twice during the season, he observes the state of the field and decides on a possible treatment with fungicides. Later, he observes the state of the field to decide on the booking of machinery for the harvest. Figure 10.12 shows an influence diagram for his decision problem.

To make an advance booking of machinery and for booking plane tickets for his summer vacation, he wishes to know the time of harvest on which he may eventually decide.

Based on the influence diagram an optimal strategy is determined, and the policy network is constructed (see Figure 10.13).

From the policy network he can read the probabilities of his future decision as to the time of harvest. After the first observation and decision, he may enter this as evidence and now get a new probability distribution for the optimal time of harvesting.

Policy networks can be used in other ways. Assume that you know the farmer's influence diagram and observe some of his actions. Then the policy network can give you estimates on what he may have observed or done in

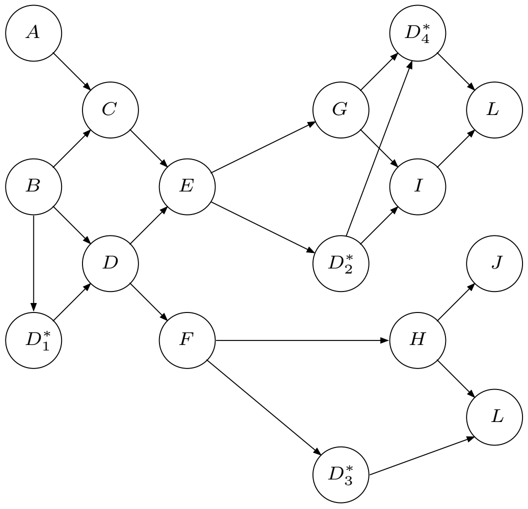

Fig. 10.11. A policy network for the influence diagram in Figure 10.5.

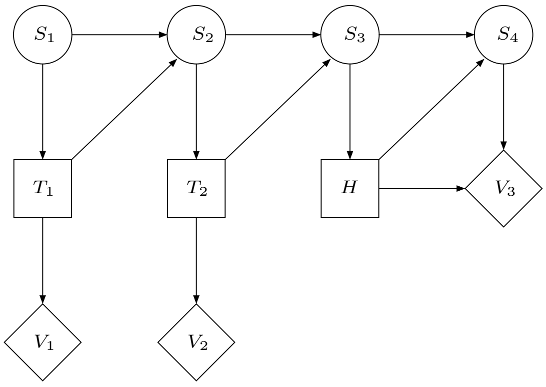

Fig. 10.12. An influence diagram for treatment and time of harvest.

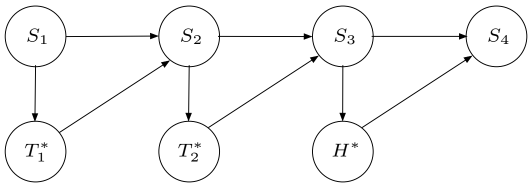

Fig. 10.13. A policy network for the influence diagram in Figure 10.12.

the past. Furthermore, policy networks can be used for analyzing the strategy proposed by the system: risk profile (what is the probability of losing \$X or going bankrupt?), probability of success (winning at least \$X), variance of the expected utility, etc.

10.3 Node Removal and Arc Reversal

In this section we present a method for solving influence diagrams by successively removing the nodes from the diagram. That is, the influence diagram is solved through the construction of a sequence of simpler and simpler influence diagrams. Actually, this method was historically the first, which worked directly on the influence diagram rather than unfolding it to a decision tree.

10.3.1 Node Removal

The method has four operations: removal of barren nodes, removal of chance nodes, removal of decision nodes, and *arc reversal.* The first three operations are rather straightforward.

Removal of barren nodes: A chance or decision node is *barren* if it has no children or if all its children are barren. Since a barren node plays no role for any decision, it can safely be removed.

Removal of chance nodes: Let the only children of the chance node C be the utility nodes $U_1, \ldots, U_k$. Then C and the utility nodes can be removed by integrating them into one utility node (see Figure 10.14) with the utility potential

$$U^* = \sum_C P(C|\,\mathrm{pa}(C)) \left[\sum_{i=1}^{k} U_i \right].$$

Removal of decision nodes: Let the only children of the decision node D be the utility nodes $U_1, \ldots, U_k$. Assume that all parents of $U_1, \ldots, U_k$ are known at the time of deciding on D. Then the optimal policy for D is

$$\delta_D = \arg\max_D \left(\sum_{i=1}^{k} U_i \right),$$

and D and $U_1, \ldots, U_k$ can be removed by substituting them with a new utility node having the potential

$$U^* = \max_D \left(\sum_{i=1}^{k} U_i \right).$$

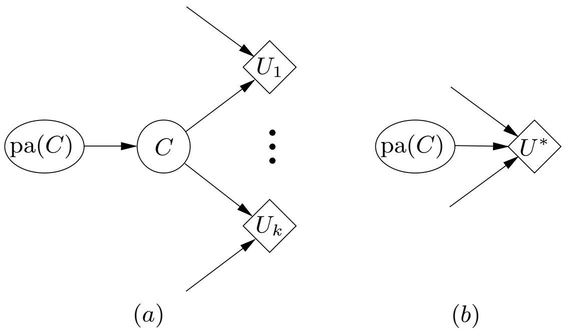

Fig. 10.14. (a) C has only utilities as children. (b) The result of removing C.

10.3.2 Arc Reversal

Consider now the influence diagram in Figure 10.15 (a). None of the removal operations can be applied. However, by applying Bayes' rule we can reverse the arrow from A to B, and now A can be removed.

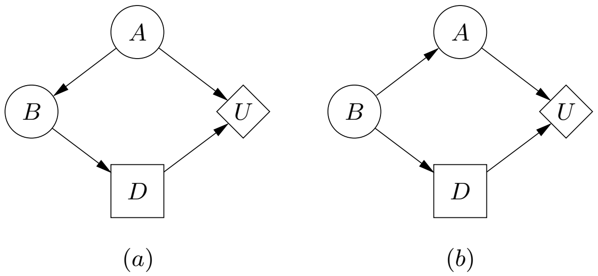

Fig. 10.15. (a) An influence diagram, where no nodes can be removed. (b) The arc has been reversed, and A can now be removed.

To generalize this operation, consider the node A with parents $C, \dots, D$, and B with parents A and $E, \dots, F$ (see Figure 10.16 (a)). Assume further that there is no other directed path between A and B.

Now, if the arc from A to B is reversed and the two nodes are given the same parents (see Figure 10.16 (b)), then all d-separation properties in the resulting Bayesian network also hold in the initial network (see Exercise 10.12).

Therefore, the resulting network can represent the probability distribution from the initial network. It is just a question of determining the new conditional probabilities. We substitute the potentials $P(A \mid C, \dots, D)$ and $P(B \mid A, E, \dots, F)$ with the potentials $P(A \mid B, C, \dots, D, E, \dots, F)$ and $P(B \mid C, \dots, D, E, \dots, F)$, and if the product of the new potentials is equal to the product of the old potentials, then the chain rule for Bayesian networks

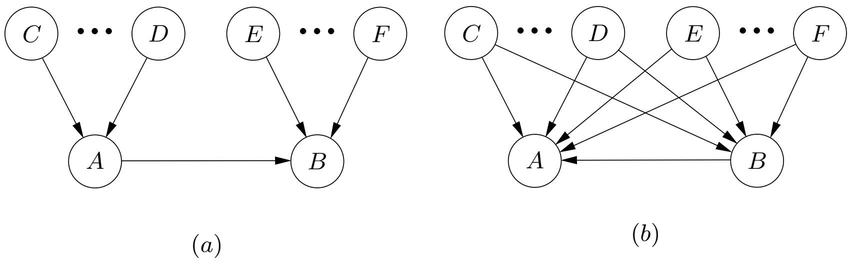

Fig. 10.16. (a) A part of a Bayesian network. (b) The arc from A to B has been reversed, and the two variables are given the same parents.

grants that the two networks represent the same probability distribution. Furthermore, we wish to use only the old potentials for the computation of the new. For this purpose we first establish the following proposition.

Proposition 10.2. *Let A be a node with parents* pa(A) *in a Bayesian network, and let X be a nonparent ancestor of A. Then X and A are d-separated given* pa(A)*.*

Proof. An active path from A to X not containing parents of A must go from A to a child of A. Since there cannot be converging connections on this path, the path must be a directed path from A to X. Since X is an ancestor of A, this would create a directed cycle; hence the path cannot be active. □

To establish the new potentials we look at $P(A, B \mid C, \ldots, D, E, \ldots, F)$. From the fundamental rule we have

$$\begin{aligned} P(A, B \mid C, \ldots, D, E, \ldots, F) =& P(B \mid A, C, \ldots, D, E, \ldots, F) \\ & P(A \mid C, \ldots, D, E, \ldots, F). \end{aligned}$$

The proposition yields that B is independent of $C, \ldots, D$ given $A, E, \ldots, F$. Since there is no directed path between A and B (other than the directed link), A is independent of $E, \ldots, F$ given $C, \ldots, D$. Hence

$$P(A, B \mid C, \ldots, D, E, \ldots, F) = P(B \mid A, E, \ldots, F) P(A \mid C, \ldots, D),$$

and this can be calculated from the potentials in the Bayesian network. Then

$$P(B \mid C, \ldots, D, E, \ldots, F) = \sum_A P(A, B \mid C, \ldots, D, E, \ldots, F),$$

and

$$P(A \mid B, C, \ldots, D, E, \ldots, F) = \frac{P(A, B \mid C, \ldots, D, E, \ldots, F)}{P(B \mid C, \ldots, D, E, \ldots, F)}.$$

Note that the product of the new potentials is equal to the product of the old potentials.

Arc reversal: Let A and B be chance nodes so that A is a parent of B and there are no other directed path from A to B. Let $C, \dots, D$ be the parents of A and let $A, E, \dots, F$ be the parents of B. Then the arc from A to B can be reversed by assigning A and B the conditional probability distributions

$$P(B \mid C, \dots, D, E, \dots, F) = \sum_A P(B \mid A, E, \dots, F) P(A \mid C, \dots, D),$$

$$P(A \mid B, C, \dots, D, E, \dots, F) = \frac{P(B \mid A, E, \dots, F) P(A \mid C, \dots, D)}{P(B \mid C, \dots, D, E, \dots, F)},$$

respectively.

10.3.3 An Example

Consider the influence diagram in Figure 10.17(a). First we remove the barren node E, and we get the influence diagram in Figure 10.17(b).

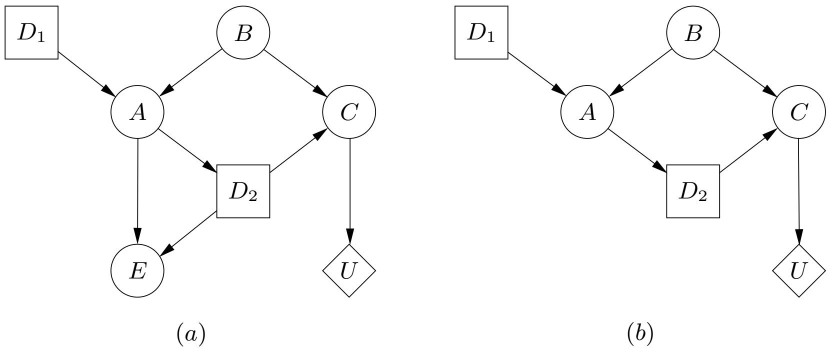

Fig. 10.17. (a) An example influence diagram. (b) The same influence diagram without barren nodes.

Next we can remove C, which has only the utility node as a child, and we get the new potential

$$U^1(D_2, B) = \sum_C U(C) P(C \mid D_2, B).$$

The resulting influence diagram is shown in Figure 10.18(a). Next, no node can be removed, and we look for an application of arc reversal. The node B cannot be removed since it has a chance variable as a child, and we fix this by arc reversal. The result is shown in Figure 10.18(b).
The new potentials are

$$P(A \mid D_1) = \sum_B P(A \mid B, D_1) P(B),$$

$$P(B \mid D_1, A) = P(A \mid B, D_1) P(B) / P(A \mid D_1).$$

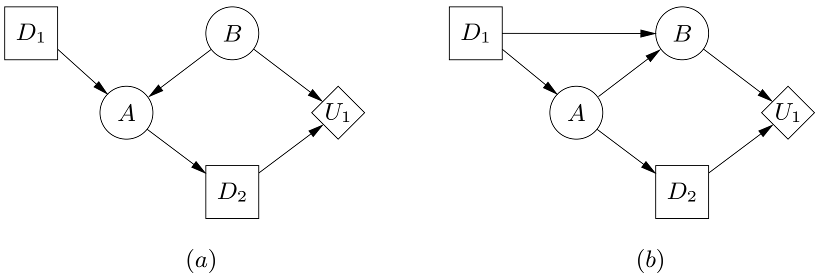

Fig. 10.18. (a) C has been removed from Figure 10.17(a). (b) The arc from B to A has been reversed.

Now we can remove B, and we get the new utility (see Figure 10.19(a))

$$U^2(D_1, A, D_2) = \sum_B U^1(B, D_2)P(B \mid A, D_1).$$

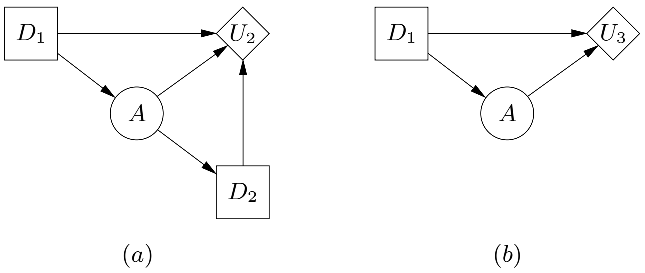

Fig. 10.19. (a) B has been removed from Figure 10.18(b). (b) D_2 has been removed.

In the influence diagram in Figure 10.19(a) we can determine the policy for D_2. We have a potential, which directly gives us the utility for each configuration of the relevant past and for each decision option. Hence the policy is achieved by determining the max-value

$$\delta_2(D_1, A) = \arg\max_{D_2} U^2(D_1, A, D_2),$$

and the maximum expected utility is

$$U^3(D_1, A) = \max_{D_2} U^2(D_1, A, D_2).$$

The result is the influence diagram in Figure 10.19(b). A chance-node removal followed by a decision-node removal does the rest.

Finally, we need to show that the four rules above are complete: all influence diagrams can be solved by successive application of the four rules. What we need to show is that if no variables can be removed, then arc reversal will bring us further. See Exercise 10.9.

10.4 Solutions to Unconstrained Influence Diagrams

A solution to an unconstrained influence diagram is an S-DAG together with optimal policies. An S-DAG containing all admissible orderings and all possible branchings after each observation can support all policies, and it could therefore be a candidate for a computational structure for the solution algorithm. However, this full S-DAG grows exponentially in the number of "holes" in the ordering, and there is a risk that it will become intractably large. Also, some nodes in the full S-DAG may never be visited by an optimal strategy, and the corresponding policy is superfluous. Therefore it is worthwhile to reduce the S-DAG under investigation.

Before presenting an algorithm for calculating optimal policies, we shall illustrate various ways of reducing the full S-DAG, while keeping it an S-DAG for an optimal strategy.

10.4.1 Minimizing the S-DAG

Consider the UID in Figure 10.20 with the full S-DAG shown in Figure 10.21; since nothing is gained by delaying a cost-free observation the observables are placed immediately after they have been released.

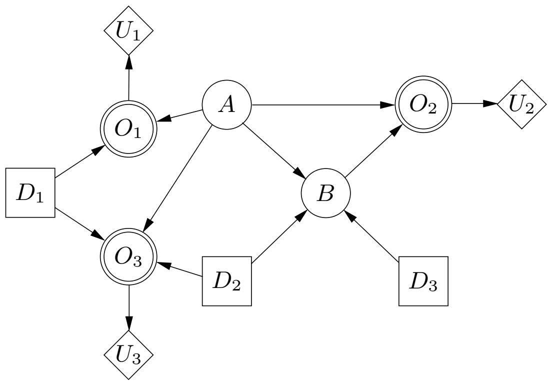

Fig. 10.20. An example UID.

In order to reduce the size of the S-DAG, you can merge paths at points where they have the same history. For example, the upper path in Figure 10.21 $D_1 - O_1 - D_2 \cdots$ shares history with the path $D_2 - D_1 - O_1 \cdots$, and from that

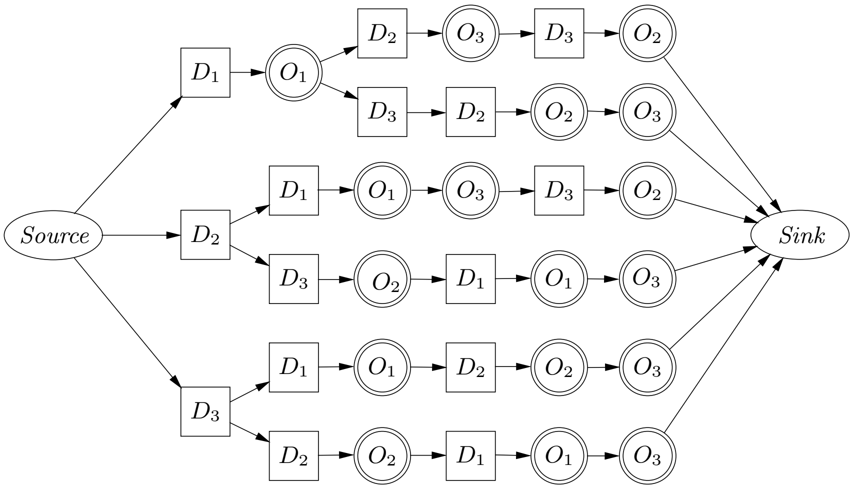

Fig. 10.21. The full S-DAG for the UID in Figure 10.20.

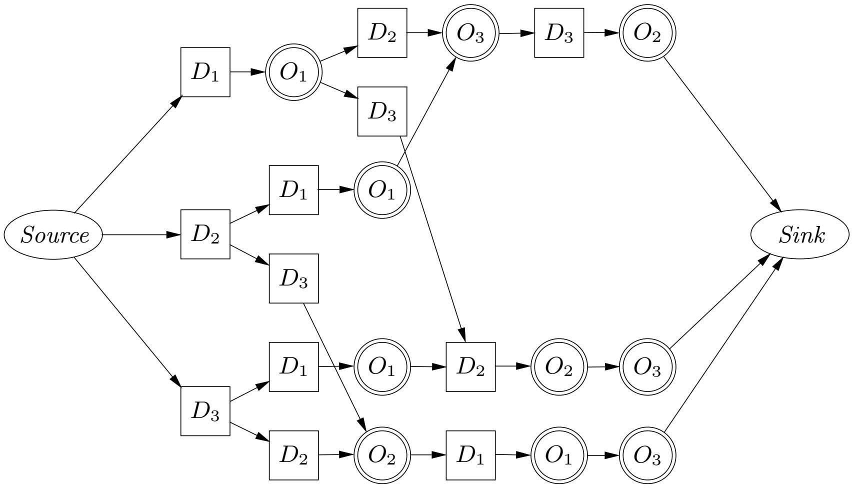

Fig. 10.22. The result of merging paths in the S-DAG from Figure 10.21.

point on, they follow the same routes. The result of merging paths according to this principle is shown in Figure 10.22.

Next, consider the path $D_2 - D_1 - O_1 \cdots$. Since the two decisions D_2 and D_1 can be swapped without changing the expected utilities, the path $D_1 - D_2 - O_1 \cdots$ will have the same expected utility as $D_2 - D_1 - O_1 \cdots$. However, on this path, the observation O_1 is not taken as soon as it has been released, and we say that O_1 is misplaced. Moving O_1 to the other side of D_2

cannot decrease the expected utilities, and we get the path $D_1 - O_1 - D_2 \cdots$. The conclusion is that the path $D_2 - D_1 - O_1 \cdots$ can never be better than $D_1 - O_1 - D_2 \cdots$, and it can therefore be removed from the S-DAG. We say that the path $D_1 - O_1 - D_2 \cdots$ *dominates* the path $D_2 - D_1 - O_1 \cdots$.

For the same reasons $D_1 - O_1 - D_3 \cdots$ dominates $D_3 - D_1 - O_1 \cdots$, $D_2 - D_3 - O_2 \cdots$ is the same as $D_3 - D_2 - O_2 \cdots$, $D_1 - O_1 - D_2 - O_3 - D_3 \cdots$ dominates $D_1 - O_1 - D_3 - D_2 - O_3 \cdots$. We end up with the S-DAG in Figure 10.23, and for this particular example the job is reduced to solving two different influence diagrams. The solution for the UID is then the optimal strategy of the one with highest expected utility.

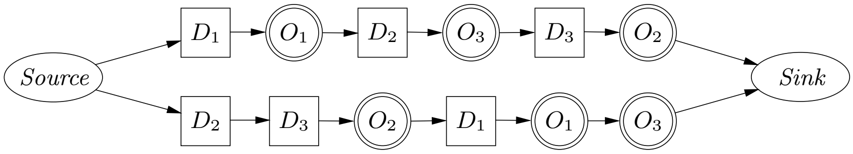

Fig. 10.23. The result of removing dominated paths from the S-DAG in Figure 10.22.

The reduction of the full S-DAG as performed above has the drawback that you start out with the full S-DAG, which may be intractably large. To avoid that, you can start from behind and build up a reduced S-DAG. The procedure is like a breadth-first search in which you go stepwise backward over the "cross section" of the S-DAG constructed so far. For the UID in Figure 10.20 you start with the sink, add all decisions that may come last, and finally add the observables released by each last decision (see Figure 10.24 (a)).

Consider the path with D_2 as the last decision. Then D_3 must be placed at some stage before (see Figure 10.24(b)). If the child of D_3 is a decision node, you can swap until you reach an observable, O. If O is not released by D_3, O is misplaced and it can be swapped with D_3. Since O_2 is the only observable released by D_3, you can move D_3 until it meets O_2, and then D_3 has passed D_2. To conclude, you can avoid D_2 as the last decision.

In general you have the following:

Proposition 10.3. *Let D be a decision node (or* Sink*) in an S-DAG, and let D_1 and D_2 be parents of D. If the set of observables released by D_1 is a subset of the set of observables released by D_2, then the path with D_2 as a parent of D can be removed without reducing the maximal expected utility.*

The proof goes along the same lines as the reasoning above in removing the path with D_2 as a parent of *Sink*.

To continue the construction of the reduced S-DAG, expand backward from D_1 and D_3. The result is shown Figure 10.25.

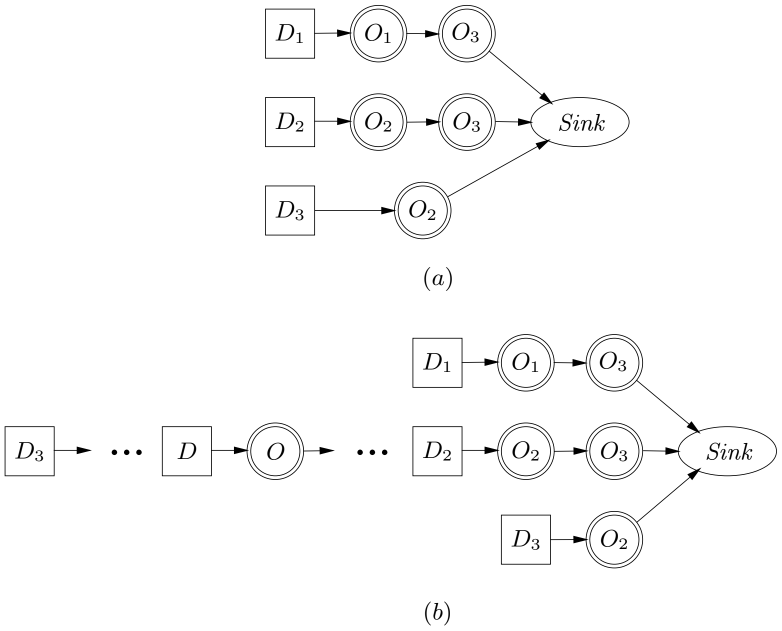

Fig. 10.24. (a) The first step in a roll-back construction of a reduced S-DAG. (b) An illustration showing why D_2 can be avoided as the last decision.

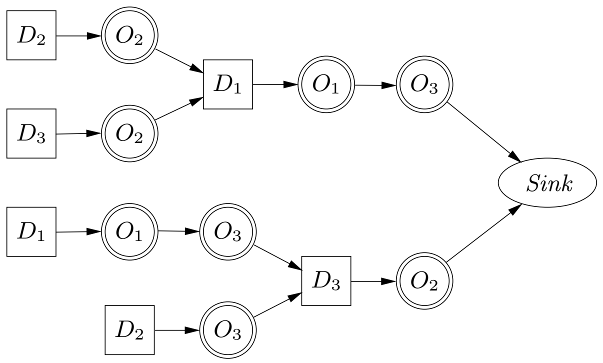

Fig. 10.25. The nodes D_1 and D_3 in Figure 10.24 are expanded backward.

Due to the proposition above, we can remove D_2 as a parent of D_1 as well as D_1 as a parent of D_3. The last expansions yield the S-DAG in Figure 10.23.

10.4.2 Determining Policies and Step Functions

A solution for a reduced S-DAG is determined in almost the same manner as for influence diagrams. We eliminate variables in reverse order; when a branching point is met, the elimination is branched out; when several paths meet, the probability potentials are the same, and the utility potentials are unified through maximization. To illustrate the method we use the UID in Figure 10.26 with the reduced S-DAG in Figure 10.27.

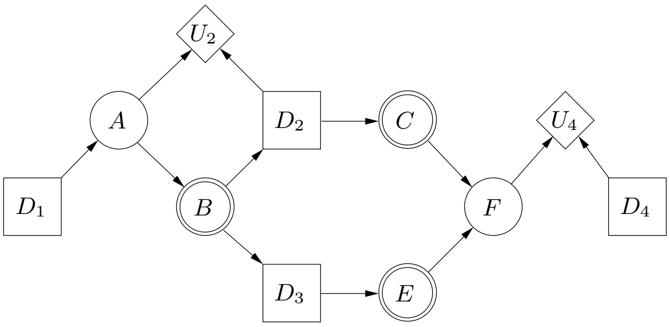

Fig. 10.26. A UID. Recall that each decision node has a hidden utility function.

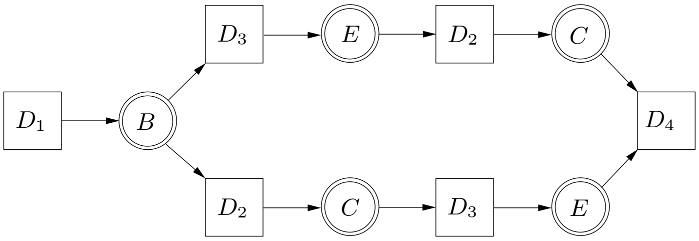

Fig. 10.27. A reduced S-DAG for the UID in Figure 10.26 (*Sink* and *Source* are ignored).

We start off with the two sets:

$$\Phi = \{P(A \mid D_1), P(B \mid A), P(C \mid D_2), P(E \mid D_3), P(F \mid C, E)\},$$
$$\Psi = \{U_1(D_1), U_2(A, D_2), U_3(D_3), U_4(F, D_4)\}.$$

First the nonobservables are eliminated. The actual variable elimination follows the same procedure as for influence diagrams (see Section 10.2). When A and F are eliminated, we get the sets

$$\begin{aligned}\Phi' &= \{P(B \mid D_1), P(C \mid D_2), P(E \mid D_3)\},\\ \Psi' &= \{U_1(D_1), U_2'(B, D_1, D_2), U_3(D_3), U_4'(C, E, D_4)\},\end{aligned}$$

where

$$\begin{aligned}P(B \mid D_1) &= \sum_A P(A \mid D_1) P(B \mid A);\\ U_2'(B, D_1, D_2) &= \frac{1}{P(B \mid D_1)} \sum_A P(A \mid D_1) P(B \mid A) U_2(A, D_2);\\ U_4'(C, E, D_4) &= \sum_F P(F \mid C, E) U_4(F, D_4).\end{aligned}$$

Note that $\sum_F P(F \mid C, E) = \mathbf{1}$. When D_4 has been eliminated we have

$$\Psi^4 = \{U_1(D_1), U_2'(B, D_1, D_2), U_3(D_3), U_4''(C, E)\},$$

where

$$\begin{aligned}U_4''(C, E) &= \max_{D_4} U_4'(C, E, D_4),\\ \delta_{D_4}(C, E) &= \arg\max_{D_4} U_4'(C, E, D_4).\end{aligned}$$

Next we branch and produce one set of potentials after elimination of C and another set after eliminating E:

$$\begin{aligned}\Phi^C &= \{P(B \mid D_1), P(E \mid D_3)\},\\ \Psi^C &= \{U_1(D_1), U_2'(B, D_1, D_2), U_3(D_3), U_4^C(E, D_2)\},\end{aligned}$$

where $U_4^C(E, D_2) = \sum_C P(C \mid D_2) U_4''(C, E)$, and

$$\begin{aligned}\Phi^E &= \{P(B \mid D_1), P(C \mid D_2)\},\\ \Psi^E &= \{U_1(D_1), U_2'(B, D_1, D_2), U_3(D_3), U_4^E(C, D_3)\},\end{aligned}$$

where $U_4^E(C, D_3) = \sum_E P(E \mid D_3) U_4''(C, E)$.

When eventually D_3 has been eliminated in the C-branch, and D_2 is eliminated in the E-branch, we have the two potential sets

$$\begin{aligned}\Phi^{Ce} &= \{P(B \mid D_1)\},\\ \Psi^{Ce} &= \{U_1(D_1), U^C(B, D_1)\};\\ \Phi^{Ec} &= \{P(B \mid D_1)\},\\ \Psi^{Ec} &= \{U_1(D_1), U^E(B, D_1)\}.\end{aligned}$$

It is no coincidence that the two probability potential sets are identical. They are both the result of sum-marginalizing the same set of variables from the same set of potentials. Since sum-marginalizations can be commuted, the two branches must give the same result. Before marginalizing B we unify the utility function sets by taking the max for each entry in the utility functions:

$$\Psi = \{U_1(D_1), \max(U^C(B, D_1), U^E(B, D_1))\}.$$

The step function is

$$\sigma(b, d_1) = \begin{cases} D_3 \text{ if } U^C(b, d_1) \geq U^E(b, d_1), \\ D_2 \text{ otherwise.} \end{cases}$$

Finally, the eliminations of B and D_1 are standard.

10.5 Decision Problems Without a Temporal Ordering: Troubleshooting

A special subclass of decision problems is that of decision problems with no temporal ordering imposed on the decisions (an extreme type of order asymmetry). An important example is troubleshooting, whereby a fault causing a device to malfunction should be identified and eliminated through a sequence of troubleshooting steps. Some steps are *repair steps*, which may or may not fix the problem; some steps are *observation steps*, which cannot fix the problem but may give indications of the causes of the problem; and some steps have repair aspects as well as observation aspects. The task is to find the cheapest strategy for sequencing the troubleshooting steps. As a first attempt you might try to model the decision problem using the unconstrained influence diagram framework, but the lack of temporal constraints will quickly cause the S-DAG to become intractably large, thereby making inference prohibitive. The car start problems of Sections 2.1.1 and 9.3.1 are examples of troubleshooting tasks.

In this section, we shall consider a solution method for decision problems with no temporal ordering by focusing solely on troubleshooting problems. In addition we will deal with pure repair steps and pure observation steps only, and we will call them *actions* and *questions*, respectively.

10.5.1 Action Sequences

First we consider a set of steps consisting of actions only. An action A_i has two possible outcomes, namely "$A_i = yes$" (the problem was fixed) and "$A_i = no$" (the action failed to fix the problem). Each action A_i has a cost $C_{A_i}(e)$, which may depend on evidence e. We sometimes use $C_i(e)$ (or C_i) as shorthand for $C_{A_i}(e)$. Because there are no questions, a *troubleshooting strategy* is a sequence of actions $\mathbf{s} = \langle A_1, \ldots, A_n \rangle$ prescribing the process of repeatedly performing the next action until an action fixes the problem or the last action has been performed.

When solving a troubleshooting problem, we have some initial evidence e and in the course of executing actions in the troubleshooting sequence $\mathbf{s} = \langle A_1, \ldots, A_n \rangle$ we collect further evidence, namely that the previous actions have failed. We let e^i denote the evidence that the first i actions have failed, and we refer to a set of failed actions as *simple evidence*. In the following, we will not mention the initial evidence explicitly.

Definition 10.3. *The* expected cost of repair *(ECR) of a troubleshooting sequence* $\mathbf{s} = \langle A_1, \ldots, A_n \rangle$ *with costs* C_i *is the mean of the costs until an action succeeds or all actions have been performed:*

$$\text{ECR}(\mathbf{s}) \equiv \sum_i \underset{i}{\text{ECR}}(\mathbf{s}),$$

where

$$\underset{i}{\text{ECR}}(\mathbf{s}) = C_i(e^{i-1})P(e^{i-1}).$$

Note that the term "expected cost of repair" may be misleading because we allow a situation in which all actions have been performed without having fixed the problem. If this happens, it will happen with the same probability regardless of the sequence, and therefore we need not estimate a cost for it. We may also extend the set of actions with a *call service* action, CS, that will fix the problem for sure. We return to this in Section 10.5.3.

Now consider two neighboring actions A_i and A_{i+1} in $\mathbf{s}$, and let $\mathbf{s}'$ be obtained from $\mathbf{s}$ by swapping the two actions. The contribution to ECR($\mathbf{s}$) from the two actions is

$$C_i(e^{i-1})P(e^{i-1}) + C_{i+1}(e^i)P(A_i = no, e^{i-1}), \tag{10.4}$$

and the contribution to ECR($\mathbf{s}'$) from the two actions is

$$C_{i+1}(e^{i-1})P(e^{i-1}) + C_i(e^{i-1}, A_{i+1} = no)P(A_{i+1} = no, e^{i-1}). \tag{10.5}$$

The difference between (10.5) and (10.4) equals ECR($\mathbf{s}'$) − ECR($\mathbf{s}$), so we get

$$\begin{aligned}\text{ECR}(\mathbf{s}') - \text{ECR}(\mathbf{s}) = P(e^{i-1}) \cdot \big[&C_{i+1}(e^{i-1}) - C_i(e^{i-1}) \\ &+C_i(e^{i-1}, A_{i+1} = no)\ P(A_{i+1} = no \,|\, e^{i-1}) - C_{i+1}(e^i)\ P(A_i = no \,|\, e^{i-1})\big]\,.\end{aligned}$$

If $\mathbf{s}$ is an optimal troubleshooting sequence, we must have ECR($\mathbf{s}$) ≤ ECR($\mathbf{s}'$), and therefore

$$\begin{aligned} C_i(e^{i-1}) + &C_{i+1}(e^i)P(A_i = no \,|\, e^{i-1}) \\ &\le C_{i+1}(e^{i-1}) + C_i(e^{i-1}, A_{i+1} = no)P(A_{i+1} = no \,|\, e^{i-1}). \end{aligned} \tag{10.6}$$

If it holds that the costs are independent of the actions taken previously, (10.6) can be rewritten as

$$\frac{P(A_i = yes \,|\, e^{i-1})}{C_i} \geq \frac{P(A_{i+1} = yes \,|\, e^{i-1})}{C_{i+1}}. \tag{10.7}$$

Definition 10.4. *Let A be a repair action and e be the evidence collected so far. The* efficiency *of A is defined as*

$$\mathrm{ef}(A \,|\, e) = \frac{P(A = \mathrm{yes} \,|\, e)}{C_A(e)}.$$

The considerations above yield the following result:

Proposition 10.4. *Let* **s** *be an optimal sequence of actions for which the costs are independent of the actions taken previously. Then it must hold that*

$$\mathrm{ef}(A_i \,|\, e^{i-1}) \geq \mathrm{ef}(A_{i+1} \,|\, e^{i-1}), \text{ for all } i.$$

10.5.2 A Greedy Approach

As remarked initially, it is not feasible to solve the troubleshooting problem using, for example, the decision tree framework or the unconstrained influence diagram framework. Alternatively, you might try to solve the troubleshooting problem by doing the sequencing in a greedy fashion: always choose an action with the highest efficiency. However, Proposition 10.4 does not guarantee that this approach will yield an optimal troubleshooting sequence.

As an example, consider Figure 10.28, where there are four possible causes, C_1, C_2, C_3, and C_4, for a malfunctioning device; we assume that exactly one of the causes is present, and that the prior probabilities are 0.2, 0.25, 0.40, and 0.15, respectively. Assume that all actions are perfect and have cost 1. Then, action A_2 has the highest efficiency, and if A_2 fails, then A_1 has higher efficiency than A_3. The sequence $\langle A_2, A_1, A_3 \rangle$ has ECR $= 1.50$. However, the sequence $\langle A_3, A_1 \rangle$ has ECR $= 1.45$.

To analyze why the decreasing-efficiency approach does not guarantee an optimal sequence, let $\langle A_1, \dots, A_n \rangle$ be a sequence ordered by decreasing efficiency. If the sequence is not optimal, there must be two actions A_i and A_j, $i < j$, that in the optimal sequence are taken in reverse order. At the time at which A_i is chosen, we have

$$\frac{P(A_i = yes \,|\, e)}{C_i} \geq \frac{P(A_j = yes \,|\, e)}{C_j}.$$

In the optimal sequence, in which A_j is chosen before A_i, we have

$$\frac{P(A_i = yes \,|\, e')}{C_i} < \frac{P(A_j = yes \,|\, e')}{C_j},$$

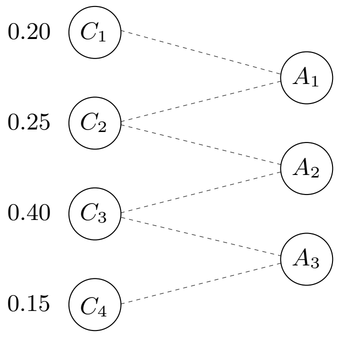

Fig. 10.28. An example of dependent actions. The C's are causes for the device failing. The A variables represent actions. An action will repair a parent if faulty. A single fault is assumed.

where e and e' are simple evidence (not involving A_i and A_j). From this we can infer that an action sequence $\langle A_1, \ldots, A_n \rangle$ is optimal if for all $i < j$ it holds that

$$\text{ef}(A_j \,|\, e) \leq \text{ef}(A_i \,|\, e),$$

where e is simple evidence (not involving A_i and A_j).

Proposition 10.5. *Consider the following assumptions:*

- *The device has n different faults $F_1, \ldots, F_n$ and n different repair actions $A_1, \ldots, A_n$.*
- *Exactly one of the faults is present.*
- *Each action has a specific probability of repair, $p_i = P(A_i = \text{yes} \,|\, F_i)$, and $P(A_i = \text{yes} \,|\, F_j) = 0$ for $i \neq j$.*
- *The cost C_i of a repair action does not depend on the performance of any previous actions.*

If these assumptions hold, then $\text{ef}(A_j) \leq \text{ef}(A_i)$ *implies that* $\text{ef}(A_j \,|\, e) \leq \text{ef}(A_i \,|\, e)$*, where e is simple evidence (not involving A_i and A_j).*

Note that we do not assume the repair actions to be perfect. They may fail to fix a fault that they are supposed to fix.

Proof. Let A_m be an action that has failed. We calculate $P(A_i = yes \,|\, A_m = no)$ (for notational convenience, we omit mention of the current evidence e). Due to the single-fault assumption, we have $P(A_m = no \,|\, A_i = yes) = 1$. Using Bayes' rule, we get

$$\begin{aligned} P(A_i = yes \,|\, A_m = no) &= \frac{P(A_m = no \,|\, A_i = yes) P(A_i = yes)}{P(A_m = no)} \\ &= \frac{P(A_i = yes)}{P(A_m = no)}. \end{aligned}$$

In other words, $P(A_m = no)$ is a normalizing constant for the remaining actions, and the relative order of efficiencies is preserved. □

Example 10.2 (Expansion of Example 9.2). On a cold and wet morning, my car will not start. Moisture may have affected the ignition system or the carburetor, the spark plugs may be dirty, there may be a lack of fuel, or there may be some other fault that I cannot fix myself.

Table 10.3 gives the initial probabilities and costs for the various causes. Because my car started yesterday evening, I assume that exactly one of the causes is present. I have one repair action for each possible cause, but the actions may not be perfect. The measure of precision is the probability of success given that the cause is present. Table 10.3 gives the precision and time requirement of the various actions.

	SP	*IS*	*Carb*	*Fu*	*Other*
Cost	4 min.	2 min.	3 min.	1 min.	n.a.
Prob.	0.3	0.1	0.1	0.1	0.4
Prec.	0.8	0.7	0.6	0.95	n.a.

Table 10.3. Initial probabilities of the causes, precision, and cost in terms of minutes for the various repair actions.

The efficiencies are calculated as

$$\begin{aligned}
\mathrm{ef}(SP) &= \frac{0.3 \cdot 0.8}{4} = 0.060,\\
\mathrm{ef}(IS) &= \frac{0.1 \cdot 0.7}{2} = 0.035,\\
\mathrm{ef}(Carb) &= \frac{0.1 \cdot 0.6}{3} = 0.02,\\
\mathrm{ef}(Fu) &= \frac{0.1 \cdot 0.95}{1} = 0.095;
\end{aligned}$$

hence I should start with *Fu*. Assume now that *Fu* did not solve the problem. By updating the efficiencies of the remaining repair actions (as in the proof above), we get

$$\begin{aligned}
\mathrm{ef}(SP) &= \frac{0.3 \cdot 0.8}{(1 - 0.1 \cdot 0.95) \cdot 4} = 0.066;\\
\mathrm{ef}(IS) &= \frac{0.1 \cdot 0.7}{(1 - 0.1 \cdot 0.95) \cdot 2} = 0.039;\\
\mathrm{ef}(Carb) &= \frac{0.1 \cdot 0.6}{(1 - 0.1 \cdot 0.95) \cdot 3} = 0.022,
\end{aligned}$$

which specify the same sequence of the remaining actions as before the update.

The following theorem concludes the considerations.

Theorem 10.3. *Let* $\mathbf{s} = \langle A_1, \ldots, A_n \rangle$ *be an action sequence for a troubleshooting problem fulfilling the conditions in Proposition 10.5. Assume that* $\mathbf{s}$ *is ordered according to decreasing initial efficiencies. Then* $\mathbf{s}$ *is an optimal action sequence and*

$$\mathrm{ECR}(\mathbf{s}) = \sum_{i=1}^{n} C_i \left(1 - \sum_{j=1}^{i-1} p_j \right). \tag{10.8}$$

Proof. From the proof of Proposition 10.5, we have that the relative order of the efficiencies of the actions is preserved. For any action sequence $\mathbf{s}'$ that is not ordered according to $\mathrm{ef}(A_i)$, there will be a j such that $\mathrm{ef}(A_j) < \mathrm{ef}(A_{j+1})$ and therefore $\mathrm{ef}(A_j \mid e^{j-1}) < \mathrm{ef}(A_{j+1} \mid e^{j-1})$. Hence, $\mathbf{s}'$ can be improved by swapping A_j and A_{j+1}. From the definition, we have

$$\mathrm{ECR}(\mathbf{s}) = \sum_{i=1}^{n} C_i P(e^{i-1}).$$

Due to the single fault assumption, we have $P(e^{i-1}) = 1 - \sum_{j=1}^{i-1} p_j$. □

10.5.3 Call Service

The action *call service* (CS) will always solve the problem. The cost of CS is not the unknown price of fixing the device but the possible overhead of having outsiders fixing a problem you could have fixed yourself. The efficiency of CS is $1/C_{CS}$ no matter what set of actions has been performed so far.

Let $\mathbf{s} = \langle A_1, \ldots, A_n \rangle$ be an optimal action sequence resulting from a situation meeting the assumptions in Proposition 10.5. It may be that the sequence should be broken before A_n and service called. According to Proposition 10.4, CS should be performed only after an action of higher efficiency. It is a good idea to perform the CS action as soon as it has maximal efficiency. However, this is not guaranteed to be optimal. The question of finding an optimal action sequence including CS is of higher combinatorial complexity: instead of looking for a sequencing of actions each of which must eventually be performed if the other actions fail, we now look for a subset of actions and a sequencing of them. We will not go further into this problem.

10.5.4 Questions

The outcome of a question may shed light on any of the possible faults, or it may be focused on a particular fault.

The troubleshooting task is to interleave actions and questions such that the expected cost is minimal. To do so, we must analyze the value of answers to questions.

Imagine that we are in the middle of a troubleshooting sequence; we have so far gained the evidence e, and now we have the option to ask the question Q with cost C_Q. For simplicity, we assume that Q has only two outcomes, "*yes*" and "*no*." Assume that regardless of the outcome of Q, we are able to calculate the minimal expected cost of repair for the remaining sequence. Therefore, let ECR be the minimal expected cost if Q is not performed, and let $\mathrm{ECR}_{Q=yes}$ and $\mathrm{ECR}_{Q=no}$ denote the same for the outcomes "*yes*" and "*no*," respectively.

Then, the value of observing Q is

$$V(Q) = \mathrm{ECR} - \left(P(Q = yes \,|\, e) \underset{Q=yes}{\mathrm{ECR}} + P(Q = no \,|\, e) \underset{Q=no}{\mathrm{ECR}} \right), \qquad (10.9)$$

and Q is performed if and only if $V(Q) > C_Q$.

In order to determine whether to ask a question prior to an action, we must analyze all possible succeeding sequences, and if there are several actions and questions, it is in general intractable. In the future, we will also have question options to interleave.

A workable approximation is the *myopic strategy*, where it is assumed at any stage of troubleshooting that we allow questions to be asked, but in the future we allow only repair actions. In that case, the task reduces to calculating expected costs given the various outcomes of the possible questions, and the approaches from the previous sections can be used.

The Myopic Repair–Observation Strategy

The following strategy is a workable approximation to the general troubleshooting task.

Algorithm 10.2 [Myopic repair–observation strategy] *To find a myopic repair-observation strategy, do:*

```
1. Let $e :=$ "the device is not working properly".
2. **While** the device is not working properly **do**
   a) Calculate EGC (the expected cost of the greedy observationless repair
      sequence).
   b) **For** all $O$ **do**
      i. **For** all states $s$ of $O$ **do**
         A. Calculate $P(O = s \,|\, e)$.
         B. **For** all $a$ **do**
            - Calculate $p_a^s = P(a \text{ solves the problem} \,|\, O = s, e)$.
         C. Calculate $EGC^s$, the expected cost of the greedy observation-
            less repair given $O = s$.
```

ii. Calculate

$$EGC_O = c_O + \sum_s P(O = s \mid e) EGC^s.$$

c) Choose the observation or action with lowest expected greedy cost; update e according to the choice and result.

□

10.6 Solutions to Decision Problems with Unbounded Time Horizon

When solving a decision problem with an unbounded time horizon, we are looking for an optimal strategy for the decisions involved. However, as opposed to optimal strategies for bounded decision problems, an optimal strategy for an unbounded decision problem will specify the same optimal policy for all the decisions (see also Section 9.6.1). In what follows we will look at solution methods for unbounded decision problems. To keep things simple we will focus on the discounted reward model, and to simplify the exposition we shall assume that the reward received in a state is independent of the chosen action.

10.6.1 A Basic Solution

As described in Section 9.6.1 we look for a utility function U^* that specifies the value of any state s assuming that all subsequent actions maximize the expected discounted reward:

$$U^*(s) = \max_{\Delta} U^*(s, \Delta) = \max_{\Delta} \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i) \,\middle|\, \Delta, s \right].$$

Instead of calculating $U^*(s, \Delta)$ directly, it can be determined from its "step wise" specification: According to the principle of maximum expected utility we should always choose the action $\delta(s)$ that maximizes the expected utility of the subsequent states:

$$\delta(s) = \arg\max_a \sum_{s' \in \mathrm{sp}(S)} P(s' \mid s, a) U^*(s'). \tag{10.10}$$

Hence, the value U^* of the current state s is the immediate reward collected at that state plus the maximum expected discounted reward of the subsequent states:

$$U^*(s) = R(s) + \gamma \max_a \sum_{s' \in \mathrm{sp}(S)} P(s' \mid s, a) U^*(s'). \tag{10.11}$$

From equation (10.10) we see that if we can calculate the maximum expected utility U^* for each state, then we can also find the optimal policy. A way of calculating U^* is to consider the equations defined by equation (10.11) as a system of $|\mathrm{sp}(S)|$ nonlinear equations with $|\mathrm{sp}(S)|$ unknowns (corresponding to the utility of each state); the nonlinearity is due to the max operator. A solution to these equations then corresponds to the utility function U^*. Unfortunately, solving such a set of equations can be a very difficult task, and instead, iterative methods are usually applied. The two most commonly applied iterative methods are called *value iteration* and *policy iteration.*

10.6.2 Value Iteration

The idea of value iteration is to start out with an initial guess at the utility U^* for each state s, and then iteratively refine this guess. How this refinement could be done is suggested by equation (10.11): the utility of a state is determined by the immediate reward received at that state plus the maximum expected utility of all the neighboring states according to our current best guess at the utility function. To be more precise, if we let U^j denote our estimate of the utility function at step j, then we can define an updating function as

$$U^{j+1}(s) = R(s) + \max_a \sum_{s'} P(s' \mid a, s) U^j(s'). \qquad (10.12)$$

The process of updating the utilities is continued for perhaps a fixed number of iterations or until the largest change is below a certain threshold value.

Example 10.3. In the robot navigation problem in Section 9.6.1, we may set the initial guess U^0 to 0. Then the first iteration sets the utilities U^1 equal to the rewards at the corresponding positions (see Figure 10.29(a)). During the next iteration we update, say position $(2, 1)$, as

$$\begin{aligned} U^2(2,1) = R(2,1) + \gamma \max \Bigg\{ & \sum_s P(s \mid \mathit{north}, (2,1)) U^1(s)\ , \\ & \sum_s P(s \mid \mathit{east}, (2,1)) U^1(s), \\ & \sum_s P(s \mid \mathit{south}, (2,1)) U^1(s), \\ & \sum_s P(s \mid \mathit{west}, (2,1)) U^1(s) \Bigg\}. \end{aligned}$$

By setting the discount factor γ to 0.9 we get

$$\begin{aligned} U^2(2,1) = & -0.1 + 0.9 \cdot \max\{0.7 \cdot -0.1 + 0.1 \cdot 10 + 0.1 \cdot -5 + 0.1 \cdot -0.1, \\ & \qquad 0.7 \cdot 10 + 0.1 \cdot -5 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1, \\ & \qquad 0.7 \cdot -5 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1 + 0.1 \cdot 10, \\ & \qquad 0.7 \cdot 0.1 + 0.1 \cdot -0.1 + 0.1 \cdot 10 + 0.1 \cdot -5\} \\ = & -0.1 + 0.9 \cdot \max\{0.42, \mathbf{6.48}, -2.52, 0.56\} \\ = & 5.73, \end{aligned}$$

and the maximal value corresponds to going east. Similarly, for position $(2,2)$ we get

$$\begin{aligned} U^2(2,2) = & -5 + 0.9 \cdot \max\{0.7 \cdot -0.1 + 0.1 \cdot -1 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1, \\ & \qquad 0.7 \cdot -1 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1, \\ & \qquad 0.7 \cdot -0.1 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1 + 0.1 \cdot -1, \\ & \qquad 0.1 \cdot 0.1 + 0.1 \cdot -1 + 0.1 \cdot -0.1 + 0.1 \cdot -0.1\} \\ = & -5 + 0.9 \cdot \max\{-\mathbf{0.19}, -0.73, -\mathbf{0.19}, -\mathbf{0.19}\} \\ = & -5.171, \end{aligned}$$

and the optimal action is then either *north*, *south*, or *west* (ties are resolved according to the sequence *west*, *south*, *east*, and *north*).

By updating the remaining utilities in this fashion we get the utility function U^2 shown in Figure 10.29(b). Based on this utility function we can continue with the third iteration (the result is shown in Figure 10.29(c)) and so forth; the optimal strategies corresponding to U^2 and U^3 (according to equation (10.10)) are shown in Figures 10.29(d)–(e); the optimal policy for U^1 is completely random.

If we continue updating the utilities according to the procedure above, the method will eventually converge to the utility function and the strategy shown in Figure 10.30(a) and Figure 10.30(c), respectively. To see the effect of the discounting factor, Figures 10.30(b) and 10.30(d) show the utility function and the optimal strategy obtained for $\gamma = 0.1$. Observe that when the value of the discounting factor is reduced (the future becomes less significant) the robot cares less about the goal state and instead focuses on avoiding the immediate obstacles. Finally, Figure 10.31 shows the maximum $\log_2$-difference in the utilities after each iteration (using $\gamma = 0.9$), which indicates that the procedure converges exponentially fast.

The fact that value iteration converged to a solution for this particular problem is no coincidence. It can be shown that value iteration is guaranteed to converge, and the utility function that it converges to is the maximum expected discounted reward. Before we give an indication as to why value iteration exhibits these properties, we shall first state the algorithm in its general form.

Algorithm 10.3 [Value Iteration] *Let γ be the discounting factor, R the reward function, and P the transition function:*

	1	2	3
1	-0.1	-0.1	10
2	-0.1	-5	-1
3	-0.1	-0.1	-0.1

(a)

	1	2	3
1	-0.19	5.73	10
2	-0.63	-5.17	4.75
3	-0.19	-0.63	-0.27

(b)

	1	2	3
1	3.42	6.23	10
2	-0.76	-1.07	5.24
3	-0.35	-0.77	2.79

(c)

	1	2	3
1	←	→	
2	←	←	↑
3	←	←	←

(d)

	1	2	3
1	→	→	
2	↓	↑	↑
3	←	←	↑

(e)

Fig. 10.29. Figures (a), (b), and (c) show the utility functions produced during the first three updates. Figures (d) and (e) show the corresponding optimal strategies; the arrows point in the direction of maximum expected discounted reward.

```
1. Choose an $\epsilon > 0$ to regulate the stopping criterion.
2. Let $U^0$ be an initial estimate of the utility function (for example, initialized
   to zero for all states).
3. Set $i := 0$.
4. Repeat
   a) Let $i := i + 1$.
   b) For each $s \in \mathrm{sp}(S)$,
         $$U^i(s) := R(s) + \gamma \cdot \max_a \sum_{s' \in \mathrm{sp}(S)} P(s' \mid a, s) U^{i-1}(s').$$
5. Until $U^i(s) - U^{i-1}(s) < \epsilon$, for all $s \in \mathrm{sp}(S)$.
```

□

It can be shown that the updating step of the algorithm ensures that the difference between any two utility functions is guaranteed to get smaller after each update. To be more specific, if we measure the difference between two utility functions as the maximum distance between two components in the functions

	1	2	3
1	6.18	7.53	10
2	4.66	1.11	6.46
3	3.90	4.04	5.28

(a)

	1	2	3
1	−0.06	0.56	10
2	−0.16	−4.97	−0.35
3	−0.11	−0.16	−0.11

(b)

	1	2	3
1	→	→	
2	↑	↑	↑
3	↑	→	↑

(c)

	1	2	3
1	→	→	
2	↑	↑	↑
3	←	←	↓

(d)

Fig. 10.30. Figures (a) and (c) show the utility function and the optimal strategy obtained upon convergence with discounting factor $\gamma = 0.9$. Convergence was achieved after 75 iterations. Figures (b) and (d) show the situation for $\gamma = 0.1$, where convergence was achieved after 18 iterations.

$$\operatorname{dist}(U_1, U_2)_{\max} = \max_{s \in \operatorname{sp}(S)} |U_1(s) - U_2(s)|,$$

then for two utility function U_1 and U_2 we have[1]

$$\operatorname{dist}\left(U_1^{i+1}, U_2^{i+1}\right)_{\max} \leq \gamma \cdot \operatorname{dist}\left(U_1^i, U_2^i\right)_{\max}.$$

In particular, if we set U_1 equal to the true utility function U^* (the solution to equation (10.11), which does not change during updates), we have

$$\operatorname{dist}\left(U^*, U^{i+1}\right)_{\max} \leq \gamma \cdot \operatorname{dist}\left(U^*, U^i\right)_{\max}.$$

This behavior allows us to derive two important properties of the updating function:

- There is only one true utility function (see Exercise 10.19).
- The value iteration algorithm is guaranteed to converge to the true utility function.

[1] The updating function is a contraction of a metric space with contraction constant γ.

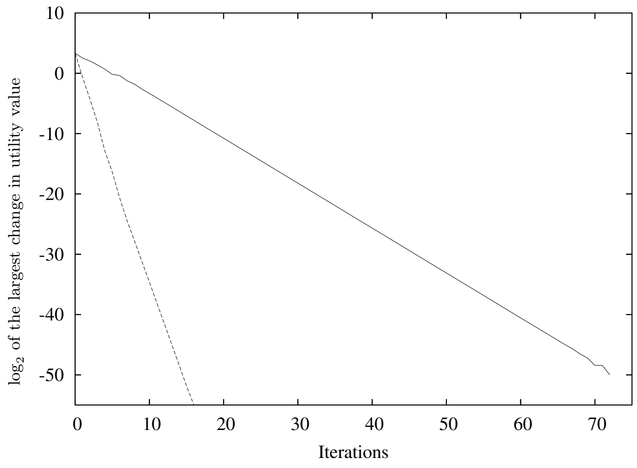

Fig. 10.31. The greatest ($\log_2$) difference in the utility values produced by a value iteration in the robot navigation problem. The solid line corresponds to the discounting factor $\gamma = 0.9$ and the dashed line corresponds to $\gamma = 0.1$.

In addition to these properties we can also find an upper bound on the number of iterations required for the distance between the true utility function and a candidate utility function to be less than ϵ. First of all, from equation (9.1) (page 329) we see that the utility of any state is bounded by $R_{\max}/(1-\gamma)$ ($R_{\max}$ is maximum *absolute* reward, $R_{\max} = \max_s |R(s)|$). Thus, for the initial iteration we have $\text{dist}\left(U^*, U^0\right)_{\max} \leq 2R_{\max}/(1-\gamma)$, and for the mth iteration we have $\text{dist}\left(U^*, U^m\right)_{\max} \leq \gamma^m \cdot 2R_{\max}/(1-\gamma)$. From the latter inequality we get

$$\text{dist}\left(U^*, U^m\right)_{\max} \leq \gamma^m \cdot 2R_{\max}/(1-\gamma) \leq \epsilon.$$

By taking the logarithm and isolating m we have $m = \log(\epsilon(1-\gamma)/2R_{\max})/\log(\gamma)$, which specifies an upper bound on the number of iterations required to achieve an error less than or equal to ϵ. In practice, however, this upper bound has a tendency to be overly conservative, and other methods have been devised to provide tighter bounds. Finally, from the equation above we can also see that the error fades away exponentially fast, but at the same time i will also quickly increase as γ approaches 1. These effects are demonstrated in Figures 10.30 and 10.31.

10.6.3 Policy Iteration

In value iteration you might say that we look for the true utility function as a means of finding an optimal policy. Another (more direct) approach, called *policy iteration*, is to perform an iterative refinement of the current best guess at an optimal policy. This method basically consists of two parts: calculate the utility function U_{Δ_i} corresponding to the current best guess Δ_i at an optimal policy [policy evaluation], and update Δ_i according to U_{Δ_i}, thereby producing an updated policy Δ_{i+1} [policy improvement]. See Figure 10.32.

$$\begin{tikzcd}
\Delta_0 \arrow[r, "\text{Policy evaluation}"'] & U_{\Delta_0} \arrow[r, "\text{Policy improvement}"'] & \Delta_1 \arrow[r, "\text{Policy evaluation}"'] & U_{\Delta_1} \arrow[r, "\text{Policy improvement}"'] & \cdots\cdots \arrow[r, "\text{Policy improvement}"'] & \Delta_m
\end{tikzcd}$$

Fig. 10.32. Policy iteration alternates between two steps: policy evaluation and policy improvement.

The idea of policy improvement is to improve our current best guess at the optimal policy Δ_i by beginning in a single state s and finding the action that maximizes the expected utility for that state assuming that the current policy is optimal for all the other states:

$$\Delta_{i+1}(s) := \arg\max_{a} \sum_{s' \in \mathrm{sp}(S)} P(s' \mid a, s) U_{\Delta_i}(s').$$

That is, we can think of policy improvement as an updating procedure for Δ_i based on a one step look-a-head according to the utility function for Δ_i:

The utility function U_{Δ_i} used in policy improvement is found during policy evaluation, where the basic task is to calculate the expected discounted reward of following the strategy Δ_i for each state s:

$$U_{\Delta_i}(s) = R(s) + \gamma \sum_{s' \in \mathrm{sp}(S)} P(s' \mid \Delta_i(s), s) U_{\Delta_i}(s').$$

Since we are working with a fixed strategy, this equation does not involve a max-operator (as opposed to our initial specification of the problem in equation (10.11)) and the expression is therefore linear in the utilities. This also means that we can calculate the utility function for a specific strategy by treating it as a linear programming problem:

$$U_\Delta(s_1) = R(s_1) + \gamma \sum_{j=1}^{n} P(s_j \mid \Delta(s_1), s_1) U(s_j),$$
$$U_\Delta(s_2) = R(s_2) + \gamma \sum_{j=1}^{n} P(s_j \mid \Delta(s_2), s_2) U(s_j),$$
$$\vdots$$
$$U_\Delta(s_n) = R(s_n) + \gamma \sum_{j=1}^{n} P(s_j \mid \Delta(s_n), s_n) U(s_j),$$

consisting of n linear equations and n unknowns. For our robot navigation problem, n corresponds to the number of possible world positions. When the state space is small, this programming problem does not introduce any difficulties, but for larger state spaces it may be too time-consuming. Instead, we can go for an approximate solution to this problem using value iteration. In this case the time complexity can be controlled by specifying a suitable termination criterion (a value for ϵ) and then using the upper bound on the number of value iterations required to reach ϵ.

In general, the policy iteration method can be stated as follows:

Algorithm 10.4 [Policy iteration]

```
1. Let $\Delta_0$ be an initial randomly chosen policy.
2. Set $i := 0$.
3. **Repeat**
   a) Find the utility function $U_{\Delta_i}$ corresponding to the policy $\Delta_i$ [policy evaluation].
   b) Let $i := i + 1$.
   c) **For** each $s \in \mathrm{sp}(S)$
         $\Delta_i(s) := \arg\max_a \sum_{s' \in \mathrm{sp}(S)} P(s' \mid a, s) U_{\Delta_{i-1}}(s')$ [policy improvement].
4. **Until** $\Delta_i = \Delta_{i-1}$.
```

□

The algorithm terminates when the current policy is not changed during an iteration. This also implies that the utility function U_{Δ_m} for the final policy Δ_m is the same as the utility function for the policy Δ_{m-1} found in the previous iteration, since they are both solutions to the same system of linear equations. Hence, U_{Δ_m} is a solution to equation (10.11):

$$\begin{aligned} U_{\Delta_m}(S) &= R(S) + \gamma \sum_{S'} P(S' \mid \Delta_m(S), s_0) U_{\Delta_m}(S') \\ &= R(S) + \gamma \sum_{S'} P(S' \mid \Delta_m(S), s_0) U_{\Delta_{m-1}}(S') \\ &= R(S) + \gamma \max_a \sum_{S'} P(S' \mid a, s_0) U_{\Delta_{m-1}}(S') \\ &= R(S) + \gamma \max_a \sum_{S'} P(S' \mid a, s_0) U_{\Delta_m}(S'). \end{aligned}$$

Since this solution is unique (see Section 10.6.2), we know that the policy returned by policy iteration is also an optimal policy.

10.6.4 Solving Partially Observable Markov Decision Processes*

As stated in Section 9.6.2, there is a fundamental difference between an optimal policy for an MDP and an optimal policy for a POMDP: an optimal policy for an MDP specifies an action for each possible state of the world, but an optimal policy for a POMDP specifies an action for each possible belief that we may have about the state of the world. A belief at step i corresponds to a probability distribution $P(S_i \mid d_1, o_1, \dots, d_{i-1}, o_i)$, which summarizes the relevant information from the past (lowercase letters are used to denote specific observations and decisions). This means that $P(S_i \mid d_1, o_1, \dots, d_{i-1}, o_i)$, our belief at step i, plays the same role as a state in an MDP, and this is also the reason why $P(S_i \mid d_1, o_1, \dots, d_{i-1}, o_i)$ is called the *belief state* at time i (denoted by $b(S_i)$ or just b_i). Thus, if $\mathcal{B}_i$ denotes the set of all possible belief states (of which there are infinitely many), then an optimal policy for decision D_i is a function

$$\delta_{D_i} : \mathcal{B}_i \rightarrow \mathrm{sp}(D_i)\,.$$

Since both value iteration and policy iteration for MDPs require a finite number of states, we cannot directly adopt these methods when working with POMDPs. Instead you might try to transform the POMDP into an "equivalent" MDP (see Figure 10.34), so that by solving the MDP we also obtain a solution to the original POMDP.

One possibility might be to simply construct a new finite belief space $\mathcal{B}'$ representing the original belief space $\mathcal{B}$. For example, in a POMDP with two world states, $\mathrm{sp}(S) = \{s_1, s_2\}$, we have a belief state for each probability of s_1; see Figure 10.33(a). This belief space can be partitioned into, for example, 10 equally wide intervals, $\mathcal{B}' = \{[0, 0.1), [0.1, 0.2), \dots, [0.9, 1]\}$, which can be used as the world states in an MDP representation. To complete the specification you also need $P(\mathcal{B}'_i \mid \mathcal{B}'_{i-1}, D_{i-1})$ and $U(D_i, \mathcal{B}'_i)$, both of which can be derived from the original POMDP specification. An approximate solution to the POMDP can now be found by solving the MDP representation using either value iteration or policy iteration.

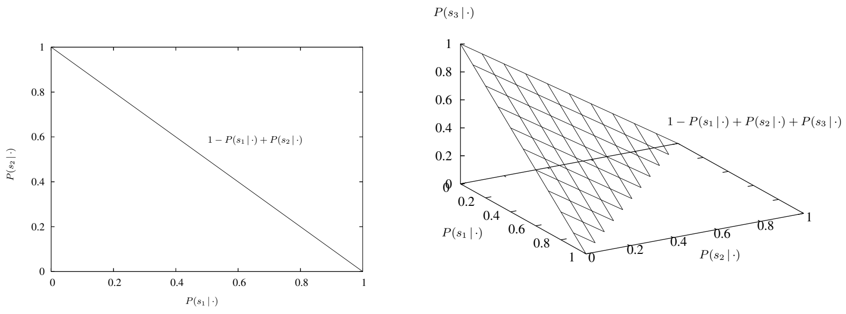

Fig. 10.33. The belief space for a POMDP with two and three world states, respectively.

Unfortunately, this partitioning/discretization procedure is infeasible for all but the smallest POMDPs, since the number of states in the MDP representation grows exponentially in the number of world states in the POMDP. Figure 10.33(b) shows a partitioning of the belief space for a POMDP with three states; with four states the belief space would be a hypercube in 4-dimensional space.

Rather than discretizing the belief space, a more common approach is to extend the MDP algorithms to infinite state spaces (see Figure 10.34). To give an idea of the procedure, let us first look at how a POMDP can be transformed into an MDP without dwelling on the potential complications of infinite state spaces.

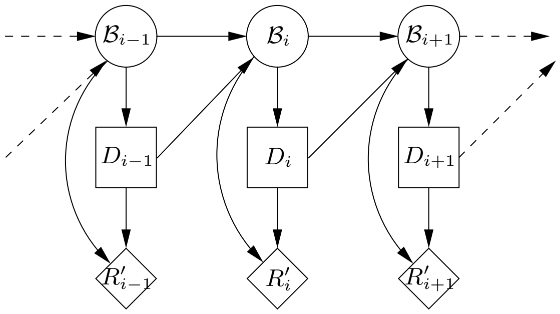

Fig. 10.34. The MDP representation of a POMDP. The state variable $\mathcal{B}_i$ contains one state for each possible belief state at step i (of which there are infinitely many).

First of all, let us start by establishing the fact that the belief state at step $i-1$ summarizes all the relevant information about the previous observations and decisions. This will also help us establish the conditional probabilities used in the MDP representation. Thus, we look for an independence relation formed by conditioning on a continuous variable. This type of conditioning is not an issue we have touched upon previously, but for the purpose of the subsequent derivations you may treat it as conditioning on a discrete variable. That is,

$$b_i = P(S_i \,|\, d_{i-1}, o_i, \text{past}(D_{i-1})) = P(S_i \,|\, d_{i-1}, o_i, b_{i-1}),$$

where $\text{past}(D_{i-1}) = (o_1, d_1, \ldots, d_{i-2}, o_{i-1})$ denotes all observations and decisions prior to decision D_{i-1}. By Bayes' rule we have that

$$\begin{aligned} &P(S_i \,|\, d_{i-1}, o_i, \text{past}(D_{i-1})) \\ &\quad = \frac{P(o_i \,|\, S_i, d_{i-1}, \text{past}(D_{i-1}))P(S_i \,|\, d_{i-1}, \text{past}(D_{i-1}))}{P(o_i \,|\, d_{i-1}, \text{past}(D_{i-1}))}, \end{aligned} \tag{10.13}$$

and since $P(o_i \,|\, d_{i-1}, \text{past}(D_{i-1}))$ is just a normalization constant, we get

$$\begin{aligned} &P(S_i \,|\, d_{i-1}, o_i, \text{past}(D_{i-1})) \\ &\quad \propto P(o_i \,|\, S_i, d_{i-1}, \text{past}(D_{i-1}))P(S_i \,|\, d_{i-1}, \text{past}(D_{i-1})). \end{aligned}$$

The third probability can also be expressed as

$$\begin{aligned} &P(S_i \,|\, d_{i-1}, \text{past}(D_{i-1})) \\ &\quad = \sum_{S_{i-1}} P(S_i \,|\, d_{i-1}, \text{past}(D_{i-1}), S_{i-1})P(S_{i-1} \,|\, d_{i-1}, \text{past}(D_{i-1})). \end{aligned}$$

Since S_{i-1} is independent of d_{i-1}, and S_i is independent of $\text{past}(D_{i-1})$ given S_{i-1} (check the d-separation properties in the model) the above expression simplifies to

$$P(S_i \,|\, d_{i-1}, \text{past}(D_{i-1})) = \sum_{S_{i-1}} P(S_i \,|\, d_{i-1}, S_{i-1})P(S_{i-1} \,|\, \text{past}(D_{i-1})).$$

By also exploiting that $P(o_i \,|\, S_i, d_{i-1}, \text{past}(D_{i-1})) = P(o_i \,|\, S_i, d_{i-1})$, equation (10.13) can now be expressed as

$$\begin{aligned} &P(S_i \,|\, d_{i-1}, o_i, \text{past}(D_{i-1})) \\ &\quad \propto P(o_i \,|\, S_i, d_{i-1}) \sum_{S_{i-1}} P(S_i \,|\, d_{i-1}, S_{i-1})P(S_{i-1} \,|\, \text{past}(D_{i-1})). \end{aligned}$$

Since $P(S_{i-1} \,|\, \text{past}(D_{i-1}))$ is the belief state, b_{i-1}, for step $i-1$ we end up with

$$\begin{aligned} b_i &= P(S_i \mid d_{i-1}, o_i, \text{past}(D_{i-1})) \\ &\propto P(o_i \mid S_i, d_{i-1}) \sum_{S_{i-1}} P(S_i \mid d_{i-1}, S_{i-1}) b(S_{i-1}), \end{aligned} \tag{10.14}$$

where the right-hand side of the expression does not depend on the past observations and decisions given the previous belief state $b(S_{i-1})$. We can therefore write

$$b(S_i) = P(S_i \mid d_{i-1}, o_i, \text{past}(D_{i-1})) = P(S_i \mid d_{i-1}, o_i, b(S_{i-1})).$$

It should also be noted that in equation (10.14) we have that $P(o_i \mid S_i, d_{i-1})$ is the observation function and $P(S_i \mid d_{i-1}, S_{i-1})$ is the transition function. Hence, equation (10.14) also provides a way to update our belief state based on the prior belief state, the decision d_{i-1}, and the observation o_i. This updated belief state corresponds to the observation of b_i.

Now, going back to our initial goal of describing the transformation of the POMDP model to the MDP model in Figure 10.34, we need to specify the transition function $P(b_i \mid b_{i-1}, d_{i-1})$ and the reward function $R'_i(b_i, D_i)$. The specification should ensure that the two models become equivalent, meaning that an optimal solution for one of the models is also an optimal solution for the other model.

The transition function $P(b_i \mid d_{i-1}, b_{i-1})$ can be expressed as

$$P(b_i \mid d_{i-1}, b_{i-1}) = \sum_{O_i} P(b_i \mid d_{i-1}, b_{i-1}, O_i) P(O_i \mid d_{i-1}, b_{i-1}), \tag{10.15}$$

where the probability $P(O_i \mid d_{i-1}, b_{i-1})$ corresponds to the normalization constant in equation (10.13) and can be calculated as

$$P(O_i \mid d_{i-1}, b_{i-1}) = \sum_{S_i} P(O_i \mid S_i, d_{i-1}) \sum_{S_{i-1}} P(S_i \mid d_{i-1}, S_{i-1}) b(S_{i-1}).$$

Again, the expression depends only on the observation function, the transition function, and the previous belief state. The function $P(b_i \mid d_{i-1}, b_{i-1}, o_i)$ is simply an indicator function defined as

$$P(b_i \mid d_{i-1}, b_{i-1}, o_i) = \begin{cases} 1 & \text{if } b(S_i) = P(S_i \mid d_{i-1}, o_i, b_{i-1}), \\ 0 & \text{otherwise.} \end{cases}$$

Next, we also have to specify the reward function $R'(b(S_i, d_i)$. Fortunately, this function can simply be calculated as

$$R'(b_i, D_i) = \sum_{S_i} R(S_i, d_i) b(S_i). \tag{10.16}$$

Thus, equation (10.15) together with equation (10.16) provides a complete specification of the transformed POMDP, and equation (10.14) describes how to find the observed belief state at each time step.

The final part is now to solve the MDP. However, we cannot immediately apply the algorithms described in the previous sections, since they work only on MDPs with finite state spaces. Instead these algorithms have to be modified to work with continuous MDPs. The overall approach is to partition the space of belief functions into regions, where each region is associated with a particular strategy and a corresponding linear utility function. A more thorough description of the algorithm is outside the scope of the present book.

10.7 Limited Memory Influence Diagrams

The major complexity problem for influence diagrams is that the relevant past for a policy may be intractably large. A way of addressing this problem is to restrict memory. This restriction can be introduced in the form of history variables or information blocking as described in Section 10.1. Another way is to pinpoint explicitly what is remembered when a decision is taken. That is, the no-forgetting assumption in interpreting an influence diagram is dropped, and instead memory is represented directly by information links.

Assume that for the fishing example in Figure 9.23 we add the restriction that we (the EU politicians) remember only last year's decision, but we can recall the T-observations up to two years back. This can be represented by the model in Figure 10.35.

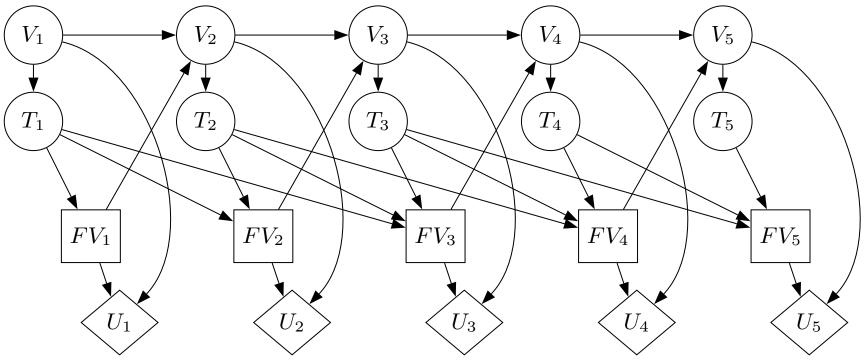

Fig. 10.35. Figure 9.23 modified to represent limited memory. Absent information arcs mean that the information is not remembered.

An influence diagram with direct representation of memory is called a *limited memory influence diagram* (LIMID). To stress the difference, influence diagrams can be called *perfect recall influence diagrams.* The advantage of LIMIDs is that they allow you to work with decision policies with small domains. If the domain of a policy does not include all the variables relevant for the associated decision, then the solution to the LIMID is an approximation to a solution for the corresponding perfect recall influence diagram.

The strong junction tree method automatically constructs cliques containing domains for perfect recall policies, and it is therefore not well suited for taking advantage of the space reduction offered by LIMIDs. Instead, a policy network can be used (see Section 10.2.3): substitute each decision variable D with a chance variable D^* having the same parents and children as D (we ignore that some informational parents may turn up nonrequired; see Section 11.2). A policy network representation of the LIMID in Figure 10.35 is shown in Figure 10.36.

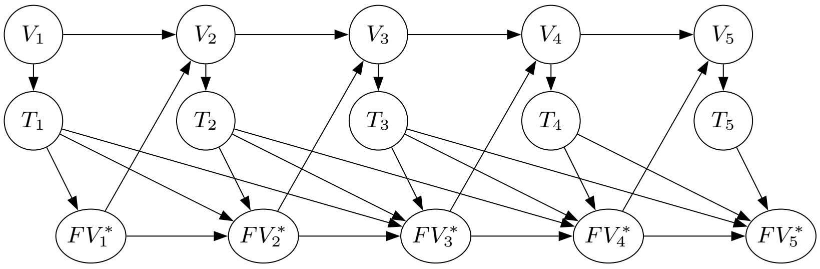

Fig. 10.36. The policy network for the LIMID in Figure 10.35.

We attach a set of initial conditional probability distributions $P_0(D^* \mid \text{pa}(D^*))$ to the D^* variables. These distributions represent our initial guess at the optimal policies of the decisions. The distributions need not be deterministic and could be chosen at random. Next, you change the policy network to a series of one-action networks and solve them as described in Section 9.1. It is natural to start with the last decision. The single-action network for the last decision in the fishing network is shown in Figure 10.37.

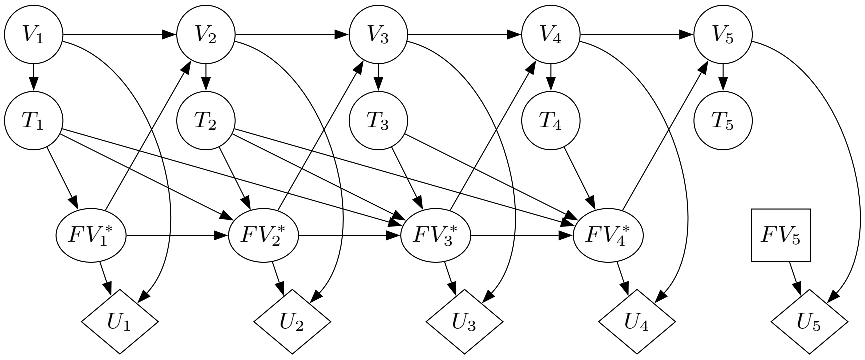

Fig. 10.37. The single-action network for the last decision in Figure 10.36.

To establish an optimal policy for FV_5 you need $P(V_5 \mid FV_4^*, T_5, T_4, T_3)$. To find this probability you can use any inference method for the underlying Bayesian network; there are no constraints on the elimination order.

Next, having found a new policy $\delta_{FV_5}(FV_4^*, T_5, T_4, T_3)$ for FV_5 you substitute the initially specified potential $P_0(FV_5^* \mid FV_4^*, T_5, T_4, T_3)$ with a chance variable representation of δ_{FV_5}:

$$P_1(FV_5^* = v \mid FV_4^*, T_5, T_4, T_3) = \begin{cases} 1 & \text{if } \delta_{FV_5}(FV_4^*, T_5, T_4, T_3) = v, \\ 0 & \text{otherwise,} \end{cases}$$

and construct the single-action network for FV_4. See Figure 10.38.

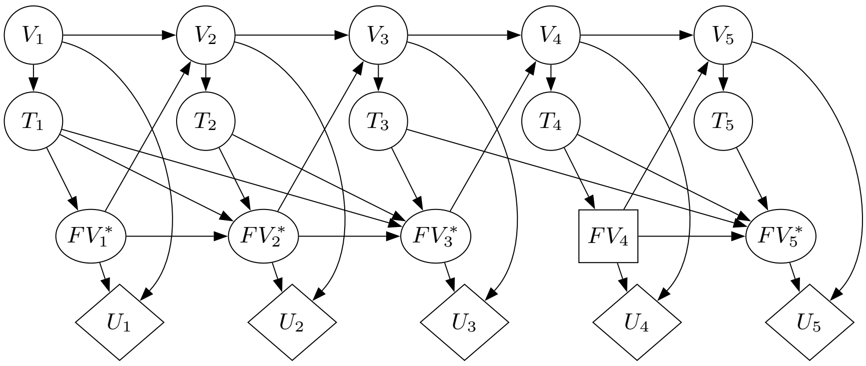

Fig. 10.38. A single-action network for FV_4.

To find a new policy for FV_4 we look for $\mathrm{EU}(FV_4 \mid FV_3, T_4, T_3, T_2)$, which is the sum of the expectations for U_4 and U_5. This requires the calculation of $P(FV_5^*, V_5 \mid FV_4, FV_3^*, T_4, T_3, T_2)$ and $P(V_4 \mid FV_4, FV_3^*, T_4, T_3, T_2)$, where the former joint probability can be found using, for example, variable propagation (see Section 5.2). Continue to FV_3 and down to FV_1.

Now, the initial policies for FV_1, FV_2, FV_3, and FV_4 were used in determining a new policy for FV_5. These initial policies also had an impact on $P(FV_5^*, V_5 \mid FV_4, FV_3^*, T_4, T_3, T_2)$ and $P(V_4 \mid FV_4, FV_3^*, T_4, T_3, T_2)$, and you need to repeat the process based on the new policies. That is, the procedure, called *single policy updating*, is iterative, and from the description above we see that it is closely related to policy iteration for MDPs. It can be shown that the procedure converges, and that it converges to an optimal strategy for the LIMID. However, this need not be an optimal strategy for the perfect recall influence diagram, and it is an issue of research to establish bounds on the distance between the LIMID optimal strategy and the perfect recall optimal strategy.

Algorithm 10.5 [Single policy updating] *Let I be a LIMID with decision variables $D_1, \dots, D_n$, and let I' be a policy network for I, where the decision variables are represented by the chance variables $D_1^*, \dots, D_n^*$.*

```
1. Let $P_0(D_j^* \mid \mathrm{pa}(D_j^*))$ be a randomly chosen initial probability distribution
   for $D_j^*$, $1 \leq j \leq n$, in $I'$.
2. Let $i := 1$.
3. Repeat
   a) For $j := n$ to $1$
      i. Let $\mathcal{U}_{D_j}$ be the utility descendants of $D_j$.[2]
      ii. Calculate a policy for $D_j$:
          $\delta_{D_j}(\mathrm{pa}(D_j))^i = \arg\max_{D_j} \sum_{U \in \mathcal{U}_{D_j}} \sum_{\mathrm{pa}(U) \setminus \mathrm{fa}(D_j)} P(\mathrm{pa}(U) \setminus \mathrm{fa}(D_j) \mid \mathrm{fa}(D_j)) U(\mathrm{pa}(U))$.
      iii. Replace $P_{i-1}(D_j^* \mid \mathrm{pa}(D_j^*))$ in $I'$ with
          $P_i(D_j^* = d \mid \mathrm{pa}(D_j^*)) = \begin{cases} 1 & \text{if } \delta_{D_j}^i(\mathrm{pa}(D_j^*)) = d, \\ 0 & \text{otherwise.} \end{cases}$
   b) Set $i := i + 1$.
4. Until convergence.
```

□

The repeated construction of single-action networks and variable propagation can be performed in a unified framework saving a large number of repetitions of the same calculations. We shall not treat this further but refer the interested reader to the literature.

10.8 Summary

The Chain Rule for Influence Diagrams

Let ID be an influence diagram with universe $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D$. Then

$$P(\mathcal{U}_C \mid \mathcal{U}_D) = \prod_{X \in \mathcal{U}_C} P(X \mid pa(X)).$$

The Expected Utility and an Optimal Strategy

Let the temporal order of the variables in $\mathcal{U}$ be described as $\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n$ and let $V = \sum_i V_i$. Then

(i) an optimal policy for D_i is

$$\delta_i(\mathcal{I}_0, D_1, \ldots, \mathcal{I}_{i-1}) = \arg\max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \ldots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \mid \mathcal{U}_D) V,$$

[2] No other utility nodes can influence the policy for D_j. See Section 11.2.

(ii) the expected utility from following the policy δ_i (and acting optimally in the future) is

$$\rho_i(\mathcal{I}_0, D_1, \dots, \mathcal{I}_{i-1}) = \frac{1}{P(\mathcal{I}_0, \dots, \mathcal{I}_{i-1} \,|\, D_1, \dots, D_{i-1})} \max_{D_i} \sum_{\mathcal{I}_i} \max_{D_{i+1}} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D) V,$$

and the strategy for ID consisting of an optimal policy for each decision yields the maximum expected utility

$$\mathrm{MEU}(ID) = \sum_{\mathcal{I}_0} \max_{D_1} \sum_{\mathcal{I}_1} \max_{D_2} \cdots \max_{D_n} \sum_{\mathcal{I}_n} P(\mathcal{U}_C \,|\, \mathcal{U}_D) V.$$

Variable Elimination for Influence Diagrams

The influence diagram is solved by repeatedly eliminating the variables in reverse temporal order. When eliminating a variable, you work with two sets of potentials: Φ, the set of probability potentials; Ψ, the set of utility potentials. When a variable X is eliminated, the potential sets are modified in the following way:

1.
$$\Phi_X := \{\phi \in \Phi \,|\, X \in \mathrm{dom}\,(\phi)\};$$
$$\psi_X := \{\phi \in \Psi \,|\, X \in \mathrm{dom}\,(\phi)\}.$$

2. If X is a chance variable, then
$$\phi_X := \sum_X \prod \Phi_X;$$
$$\psi_X := \sum_X \prod \Phi_X \left(\sum \Psi_X\right).$$

   If X is a decision variable, then[3]
$$\phi_X := \max_X \prod \Phi_X;$$
$$\psi_X := \max_X \prod \Phi_X \left(\sum \Psi_X\right).$$

3.
$$\Phi := \left(\Phi \setminus \Phi_X\right) \cup \{\phi_X\},$$
$$\Psi := \left(\Psi \setminus \Psi_X\right) \cup \left\{\frac{\psi_X}{\phi_X}\right\}.$$

These calculations can also be organized in a strong junction tree for the influence diagram. A strong junction tree is produced by eliminating the variables in reverse temporal order.

[3] When X is a decision variable, Φ_x is a constant function over X.

Policy Networks

Let D be a decision variable with policy $\delta_D(\mathrm{req}(D))$. The chance-variable representation of D is the result of the following construction: Substitute D with a chance variable D^* with parents $\mathrm{req}(D)$. The conditional probability potential $P(D^* \mid \mathrm{req}(D))$ is

$$P(d|\bar{r}) = \begin{cases} 1 & \text{if } \delta_D(\bar{r}) = d, \\ 0 & \text{otherwise.} \end{cases}$$

Let ID be an influence diagram over $\mathcal{U} = \mathcal{U}_C \cup U_D$. A policy network for ID (denoted by ID^*) is a Bayesian network over $\mathcal{U} = \mathcal{U}_C \cup \mathcal{U}_D^*$ in which all decision variables D_i have been substituted with their chance-variable representations. The probability potentials from ID are kept (with D_js replaced by D_j^*).

Node Removal and Arc Reversal

The influence diagram is solved by iteratively removing nodes and reversing arcs according to the following rules:

Removal of barren nodes: A chance or decision node is *barren* if it has no children or all its children are barren. Since a barren node plays no role for any decision, it can safely be removed.

Removal of chance nodes: Let the only children of the chance node C be the utility nodes $U_1, \ldots, U_k$. Then C and the utility nodes can be removed by integrating them into one utility node with the utility potential

$$U^* = \sum_C P(C|\,\mathrm{pa}(C)) \left[\sum_{i=1}^{k} U_i \right].$$

Removal of decision nodes. Let the only children of the decision node D be the utility nodes $U_1, \ldots, U_k$. Assume that all parents of $U_1, \ldots, U_k$ are known at the time of deciding on D. Then the optimal policy for D is

$$\delta_D = \arg\max_D \left(\sum_{i=1}^{k} U_i \right),$$

and D and $U_1, \ldots, U_k$ can be removed by substituting them with a new utility node having the potential

$$U^* = \max_D \left(\sum_{i=1}^{k} U_i \right).$$

If no nodes can be removed, then arc reversals can be performed to obtain another (EU-equivalent) influence diagram in which one of the rules above can be applied.

Arc reversal: Let A and B be chance nodes such that A is a parent of B and there are no other directed paths from A to B. Let $C, \dots, D$ be the parents of A and let $A, E, \dots, F$ be the parents of B. Then the arc from A to B can be reversed by assigning A and B the conditional probability distributions

$$P(B \mid C, \dots, D, E, \dots, F) = \sum_A P(B \mid A, E, \dots, F) P(A \mid C, \dots, D),$$
$$P(A \mid B, C, \dots, D, E, \dots, F) = \frac{P(B \mid A, E, \dots, F) P(A \mid C, \dots, D)}{P(B \mid C, \dots, D, E, \dots, F)},$$

respectively.

Unconstrained Influence Diagrams

An S-DAG can be constructed from a breadth-first procedure starting at the sink: add all the decisions that may come last, and after that you add the observables released by the decisions. By exploiting the following rule we need not construct the full S-DAG:

Let D be a decision node (or *Sink*) in an S-DAG, and let D_1 and D_2 be parents of D. If the set of observables released by D_1 is a subset of the set of observables released by D_2, then the path with D_2 as a parent of D can be removed without reducing the maximal expected utility.

A solution to the UID is found using variable elimination based on the S-DAG structure.

Troubleshooting

The expected cost of repair of a troubleshooting sequence $\mathbf{s} = \langle A_1, \dots, A_n \rangle$ of repair actions is

$$\mathrm{ECR}(\mathbf{s}) = \sum_i C_i(\mathbf{e}^{i-1}) P(\mathbf{e}^{i-1}),$$

where $\mathbf{e}^j$ denotes the statement that the first j actions have failed.

For an optimal repair sequence, it holds that

$$C_i(\mathbf{e}^{i-1}) + C_{i+1}(\mathbf{e}^i) P(A_i = n \mid \mathbf{e}^{i-1}) \\ \leq C_{i+1}(\mathbf{e}^{i-1}) + C_i(\mathbf{e}^{i-1}, A_{i+1} = n) P(A_{i+1} = n \mid \mathbf{e}^{i-1}).$$

The efficiency of a repair action is

$$\mathrm{ef}(A \mid \mathbf{e}) = \frac{P(A = y \mid \mathbf{e})}{C_A(\mathbf{e})}.$$

If costs are independent of evidence, then for an optimal repair sequence it must hold that

$$\text{ef}(A_i \mid \mathbf{e}^{i-1}) \geq \text{ef}(A_{i+1} \mid \mathbf{e}^{i-1}),$$

and if for all $i < j$ it holds that

$$\text{ef}(A_j \mid \mathbf{e}) \leq \text{ef}(A_i \mid \mathbf{e})$$

for all simple evidence $\mathbf{e}$ (not involving A_i and A_j) of the type "actions $A, \ldots, B$ have failed," then the repair sequence $\langle A_1, \ldots, A_n \rangle$ is optimal (this does not necessarily hold when call service is an option).

Questions: The value of getting an answer of Q is

$$V(Q) = \text{ECR} - \sum_{s \in Q} P(Q = s \mid \mathbf{e}) \, \underset{s}{\text{ECR}},$$

where ECR_s is the expected cost of repair for an optimal sequence given evidence $\mathbf{e}$ and "$Q = s$," and ECR is the expected cost of repair for an optimal sequence not starting with Q. Because neither ECR nor ECR_s is tractable, a myopic approach is often used.

Unbounded Decision Problems

Let γ be the discounting factor, R the reward function, and P the transition function.

Value iteration:

1. Choose an $\epsilon > 0$ to regulate the stopping criterion.
2. Let U^0 be an initial estimate of the utility function (for example, initialized to zero for all states).
3. Set $i := 0$.
4. **Repeat**
   a) Let $i := i + 1$.
   b) **For** each $s \in \text{sp}(S)$
   $$U^i(s) := R(s) + \gamma \cdot \max_a \sum_{s' \, \text{sp}(S)} P(s' \mid a, s) U^{i-1}(s').$$
5. **Until** $U^i(s) - U^{i-1}(s) < \epsilon$, for all $s \in \text{sp}(S)$.

Policy iteration:

1. Let Δ_0 be some initial (randomly chosen) policy.
2. Set $i := 0$.
3. **Repeat**
   a) Find the utility function U_{Δ_i} corresponding to the policy Δ_i [policy evaluation].

```
    b) Let $i := i + 1$.
    c) **For** each $s \in \mathrm{sp}(S)$

       $$\Delta_i(s) := \arg\max_a \sum_{s' \in \mathrm{sp}(S)} P(s' \mid a, s) U_{\Delta_{i-1}}(s') \text{ [policy improvement]}.$$

4. **Until** $\Delta_i = \Delta_{i-1}$.
```

Limited Memory Influence Diagrams (LIMIDs)

The no-forgetting assumption is dropped and instead, the informational arcs specify the variables observed before a particular decision (thereby controlling the size of the policy functions). A solution can be found using the single policy updating algorithm:

Single policy updating: Let I be a LIMID with decision variables $D_1, \ldots, D_n$, and let I' be a policy network for I in which the decision variables are represented by the chance variables $D_1^*, \ldots, D_n^*$.

```
1. Let $P_0(D_j^* \mid \mathrm{pa}(D_j^*))$ be an initial probability distribution (chosen at random) for $D_j^*$, $1 \leq j \leq n$, in $I'$.
2. Let $i := 1$.
3. Repeat
    a) For $j := n$ to 1
        i. Let $\mathcal{U}_{D_j}$ be the utility descendants of $D_j$.
        ii. Calculate a policy for $D_j$:

            $$\delta_{D_j}(\mathrm{pa}(D_j))^i = \arg\max_{D_j} \sum_{U \in \mathcal{U}_{D_j}} \sum_{\mathrm{pa}(U) \setminus \mathrm{fa}(D_j)} P(\mathrm{pa}(U) \setminus \mathrm{fa}(D_j) \mid \mathrm{fa}(D_j)) U(\mathrm{pa}(U)).$$

        iii. Replace $P_{i-1}(D_j^* \mid \mathrm{pa}(D_j^*))$ in $I'$ with

            $$P_i(D_j^* = d \mid \mathrm{pa}(D_j^*)) = \begin{cases} 1 & \text{if } \delta_{D_j}^i(\mathrm{pa}(D_j^*)) = d, \\ 0 & \text{otherwise.} \end{cases}$$

    b) Set $i := i + 1$.
4. Until convergence.
```

10.9 Bibliographical Notes

Various methods for solving influence diagrams have been constructed. Olmsted (1983) and Shachter (1986) introduced arc-reversal, and Shenoy (1992), Jensen *et al.* (1994), Cowell (1994), Ndilikilikesha (1994), and Madsen and

Jensen (1999a) used elimination and direct manipulation of potentials. Cooper (1988) presents a method that works well for scenarios with one decision variable. It substitutes the decision variable and the utility variables with chance variables and uses Bayesian network propagation. Zhang (1998) exploits Cooper's method to full influence diagrams.

The solution strategy for unconstrained influence diagrams was proposed in (Jensen and Vomlelova, 2002). A solution algorithm for sequential influence diagrams can be found in (Jensen *et al.*, 2006).

Troubleshooting based on decision theory was introduced by Kalagnanam and Henrion (1990), and it was further analyzed by Heckerman *et al.* (1995a). Section 10.5 is an extension of this work. Proofs that various versions of troubleshooting are NP-complete can be found in (Vomlelová, 2003).

The main ideas involved with solving Markov decision processes through value iteration originates with Shapley (1953). Policy iteration originates with Howard (1960).

LIMIDs were proposed in (Nilsson and Lauritzen, 2000).

10.10 Exercises

Exercise 10.1. Consider the influence diagram in Figure 9.22. Is L d-separated from E given I? Find a minimal set of nodes that d-separate A from D_3.

Exercise 10.2. Consider the influence diagram DI from Figure 10.1 but without the utility node V_1. Derive the formulas for an optimal strategy.

Exercise 10.3. Prove that during variable elimination, the potential $\prod \Phi_D$ is constant over D if all the variables following D in the partial ordering have already been eliminated.

Exercise 10.4. Construct a strong junction tree for the influence diagram in Figure 9.21 and determine the domains of the policies.

Exercise 10.5. Construct strong junction trees for the influence diagrams in Figures 9.23 and 9.24. Compare the clique sizes and the domains of the policies.

Exercise 10.6. Show that any strong triangulation of the influence diagram in Figure 10.10 will place E and B in the clique where D_1 is eliminated.

Exercise 10.7. Construct a strong junction tree for the influence diagram in Figure 10.39

(i) Is D_2 required for D_3?
(ii) Is B required for D_3?

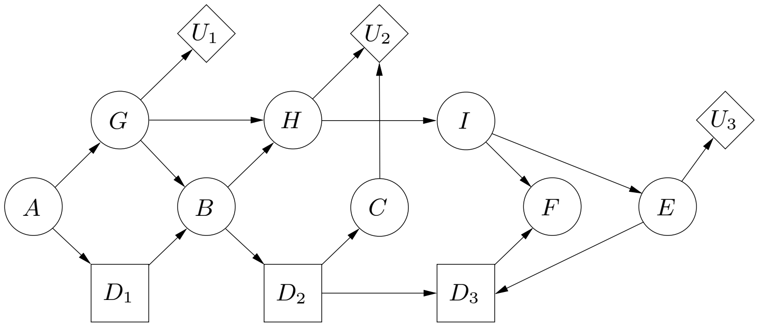

Fig. 10.39. Figure for Exercise 10.7.

(iii) Construct a join tree for the policy network and compare the size with the size of the strong junction tree.

Exercise 10.8. (i) Let $\{a_{ij}\}$ be an $n \times m$ matrix of reals. Prove that

$$\max_i \sum_j a_{ij} \leq \sum_j \max_i a_{ij}.$$

(ii) Use (i) to show that the MEU of an influence diagram will not increase by delaying an observation. (Hint: Look at the formulas for the two elimination orders.)

Exercise 10.9. Consider the arc-reversal solution method for influence diagrams, and a point where no node can be removed (because the only nodes with only utility nodes as children are decision nodes and these utility nodes have nonobservables as parents as well). To show that we can always find an arc to reverse, prove that there is at least one pair of chance nodes A and B such that A is a parent of B and there is no other directed path from A to B.

Exercise 10.10. Consider the simple influence diagram in Figure 10.40, where all variables are binary, and the probabilities for C_1 are given in Table 10.4, the probability of $C_2 = c_2$ is 0.8, and the utility functions U_1 and U_2 are given in Tables 10.5 and 10.6. Solve the influence diagram using node removal and arc reversal.

$D_1 \setminus C_2$	c_2	$\neg c_2$
d_1	0.2	0.7
$\neg d_1$	0.5	0.5

Table 10.4. $P(C_1 = c_1 \mid D_1, C_2)$.

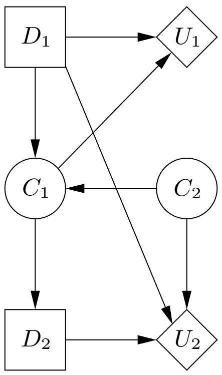

Fig. 10.40. A simple influence diagram.

$D_1 \setminus C_1$	c_1	$\neg c_1$
d_1	5	-2
$\neg d_1$	3	-10

Table 10.5. $U_1(D_1, C_1)$.

$D_1 \setminus C_2$	c_2	$\neg c_2$
d_1	$(0, 0)$	$(8, -5)$
$\neg d_1$	$(5, -1)$	$(1, 12)$

Table 10.6. $U_2(D_1, C_2, D_2)$. Entries should be interpreted as $(d_2, \neg d_2)$.

Exercise 10.11. Which steps would be carried out if the influence diagram in Figure 9.22 were solved using node removal and arc reversal? Assuming that each node has two states, what is the largest potential constructed during the solution process?

Exercise 10.12. Let I be an influence diagram, and I' be the influence diagram obtained by reversing an arc in I. Prove that if X and Y are variables d-separated by a set of variables $\mathcal{Z}$ in $\mathcal{I}'$, then X and Y are also d-separated given $\mathcal{Z}$ in I.

Exercise 10.13. Prove that when the node removal and arc reversal solution method is applied to an influence diagram, it eliminates the decision variables in an order consistent with the partial temporal ordering of the nodes in the diagram.

Exercise 10.14. Consider the UID in Figure 9.48. Construct the full S-DAG for the UID, and then reduce it as much as possible. Is the result the same when you do a roll-back construction of the S-DAG?

Exercise 10.15. Use the algorithm in Section 10.5.4 to solve the start problem in Example 9.2 (Page 293).

Exercise 10.16. ^{E} You are experiencing irregularities using your computer. There are several reasons why this can be: first, one of the programs you are running can be malfunctioning and interfering with your operating system; second, you can have attracted a virus; and third, you can have a hardware problem. Assuming that only one problem exists, the probabilities of the three problems are 0.8, 0.15, and 0.05, respectively. Your possible actions for fixing the problem are

1. Reboot the computer.
2. Run a virus removal tool.
3. Reformat your hard disk and reinstall your operating system.
4. Buy a new computer.

The costs of each option as an overall index of frustration, time usage, and money spent are 1, 2, 25, and 500, respectively. The probability of action 4 solving the problem is 1 no matter what the problem is and which other attempts to solve the problem have failed so far. Action 3 has a probability of 0.99 of fixing the problem if it is a nonhardware problem, and 0 if it is a hardware problem, no matter which other solutions that have failed previously. Action 2 solves the problem with probability 0.95 if it is a virus problem, and with probability 0 otherwise, again no matter what other solutions have unsuccessfully been tried. Finally, action 1 solves the problem with probability 1 if it is due to a malfunctioning program, and 0 otherwise, no matter what previous unsuccessful attempts at solving the problem were tried.

Formulate the above setting as a troubleshooting problem, and give an optimal sequence of repair actions. What is the expected cost of repair for the sequence?

Exercise 10.17. ^{E} Consider again the computer problem in Exercise 10.16, and assume further that you are given the option of buying a computer program that can scan the computer for hardware errors. The overall effort involved in doing this is 4. If there is a hardware error, the program has a 0.999 chance of discovering it, and there is no risk of false positives. Moreover, you are given the choice of having your computer scanned remotely on the Internet by some company for a price of 0.25. The scanning discovers a virus with a probability of 0.99 if there is one, but the scanner cannot remove it. For that you are given the option of downloading a special virus-removal program, which has a cost of 2 and which removes the identified virus with a probability of 1. Are the two offers individually worth the asking price? Are they worth the price in combination?

Exercise 10.18. Continue Example 10.3 and perform one more iteration of value iteration starting with the utility function shown in Figure 10.29(c).

Exercise 10.19. Show that there is only one true utility function representing the maximum expected discounted reward of a Markov decision process with an unbounded time horizon.

Exercise 10.20. ^{E} Consider the influence diagram in Example 10.10, but interpreted as a LIMID. Using the policies $D_1 = \neg d_1$ and $D_2 = d_2$, regardless of the state of C_1, run two iterations of policy updating.

Exercise 10.21. ^{E} Consider the LIMID in Figure 10.41, with its realization specified as in Example 10.10. Using the policies $D_1 = \neg d_1$ and $D_2 = d_2$, regardless of the states of C_1 and D_1, run two iterations of policy updating.

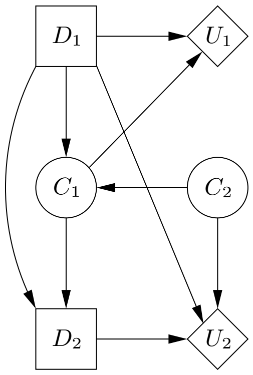

Fig. 10.41. A LIMID for Exercise 10.21.