Chapter 5

Probabilistic planning

Probabilistic planning is an extension of nondeterministic planning with exact information on the probabilities of nondeterministic events.

Exact probabilities are important because it is not just important to get things done, but to get them done efficiently, and for goals for which there is no guarantee that they are reached, it is important to reach them as likely as possible.

The introduction of probabilities complicates planning, both conceptually and computationally. Whereas in the non-probabilistic of conditional planning with partial observability it is sufficient to work in a finite discrete belief space, the introduction of probabilities makes the belief space continuous and thereby infinite. This means that there are no algorithms for doing planning, that is, there is no program that either delivers a plan (with a given property) or announces that no plans exist.

However, despite these difficulties one is forced to face, probabilities are important in many types of applications, and algorithms for probabilistic planning are therefore worth studying.

In this section we discuss a number of algorithms for probabilistic planning, starting from algorithms for the conditional planning problem with full observability. The use of probabilities allows to consider more general plan quality criteria than those that were considered in connection with non-probabilistic planning problems. A main difference is that there is no necessity to restrict to planning with the objective of reaching one of designated goal states. Instead, actions and states are associated with rewards/costs, and the objective is to maximize the rewards (or minimize costs) over the execution of a plan. This kind of problems naturally generalize to plan executions of infinite length.

5.1 Stochastic transition systems with rewards

In Section 2.1 we gave a basic definition of deterministic and nondeterministic transition systems. For expressing exact transition probabilities we need a new definition of transition systems.

A stochastic transition system consists of a finite set S of states. The actions do not just associate a set of possible successor states to each state, but a probability distribution on the set of possible successor states.

An action is a partial function from S to probability distributions over S. Partiality means that not all actions are applicable in all states. A probability distribution p is a function that maps S to real numbers $r \in [0, 1]$ so that $\sum_{s \in S} p(s) = 1.0$. The probability distribution indicates how likely each state is as a successor state of a given state.

In many types of probabilistic planning problems considered in the literature the objective is not to reach one of a set of designated goal states. Instead, the objective is to act in a way that maximizes the *rewards* or minimizes the *costs*. Planning problems with a designated set of goal states can be expressed in terms of rewards, but not vice versa.

Definition 5.1 *A* stochastic transition system with rewards *is a 4-tuple* $\langle S, A, p, R, \rangle$ *where*

- S *is a finite set of states,*
- A *is a finite set of actions,*
- p *is a partial function that maps each state* $s \in S$ *and action* $a \in A$ *to a probability distribution on* S*, and*
- $R : S \times A \rightarrow \mathcal{R}$ *is a reward function which maps each state* $s \in S$ *and action* $a \in A$ *to real number.*

A major difference to the definition of Markov decision processes [Puterman, 1994] is that p is a partial function, that is, not all states are assigned a probability distribution. This is for having a match between the definition of operators in AI planning, where not all actions are applicable in all states. Below, we will denote the set of actions applicable in a state $s \in S$ by $A(S)$. We also require that $A(s)$ is non-empty for every $s \in S$.

Notice that we have not defined initial states or a probability distribution on possible initial states: the most important algorithms find plans that reach the goals from any initial state. Clearly, when the number of states is very high and the sets of initial states are small, more efficient planning could be obtained by taking information about the set of initial states into account.

Stochastic transition systems can be described in terms of state variables and operators just like the transition systems earlier discussed in this lecture. A nondeterministic operator $\langle c, e \rangle$, as given in Definition 4.1, assigns a probability distribution corresponding to e, as given in Definition 4.2, to any state s such that $s \models c$.

5.2 Problem definition

A given plan produces infinite sequences of rewards $r_1, r_2, \ldots$. Clearly, if the planning problem has several initial states or if the actions are nondeterministic this sequence of rewards is not unique. In either case, possible plans are assessed in terms of these rewards, and there are several possibilities how good plans are defined. Because the sequences are infinite, we in general cannot simply take their sum and compare them. Instead, several other possibilities have been considered.

1. Expected total rewards over a finite horizon.

   This is a natural alternative that allows using the normal arithmetic sum of the rewards. However, there is typically no natural bound on the horizon length.

2. Expected average rewards over an infinite horizon.

   This is probably the most natural way of assessing plans. However, there are several technical complications that make average rewards difficult to use.

3. Expected discounted rewards over an infinite horizon.

   This is the most often used criterion in connection with Markov decision processes. Discounting means multiplying the ith reward by λ^{i-1} and it means that early rewards are much more important than rewards obtained much later. The discount constant λ has a value strictly between 0.0 and 1.0. The sum of the geometrically discounted rewards is finite. Like with choosing the horizon length when evaluating plans with respect to their behavior within a finite horizon, it is often difficult to say why a certain discount constant λ is used.

For the latter two infinite horizon problems there always is an optimal plan that is a mapping from states to actions, and this is the type of plan used in most of this section.

In the first case with a bounded horizon the optimal plans cannot be represented as mappings from state to actions *if it really is the case that the length of the plan execution indeed equals the horizon length*, and instead mappings from states and time points to actions are needed. This is because for example at the last stage all rewards that are obtained are from the last action. The optimal plans are therefore time-dependent. However, nothing prevents using the first stage of the finite horizon plan as a normal plan, that is, as a mapping from states to actions.

We state the probabilistic conditional planning problem in the general form. Like with non-probabilistic conditional planning, observability restrictions are expressed in terms of a set of state variables that are observable.

Definition 5.2 *A 5-tuple $\langle A, I, O, B, R\rangle$ consisting of a set A of state variables, a probability distribution I over valuations of A, a set O of operators, a reward function R, and a set $B \subseteq A$ of state variables is* a problem instance in probabilistic nondeterministic planning.

I is a set $\{\langle\phi_1, p_1\rangle, \langle\phi_2, p_2\rangle, \ldots, \langle\phi_n, p_n\rangle\}$ that expresses a probability distribution over valuations of A. We require that $\phi_i \models \neg\phi_j$ for every $\{i, j\} \subseteq \{1, \ldots, n\}$.

$R(o)$ for every $o \in O$ is a set $\{\langle\phi_1, r_1\rangle, \langle\phi_2, r_2\rangle, \ldots, \langle\phi_m, r_m\rangle\}$ that expresses the rewards obtained when o is applied: if o is applied in s and $s \models \phi_k$, then reward is r_k. We require that $\phi_i \models \neg\phi_j$ for every $\{i, j\} \subseteq \{1, \ldots, m\}$.

Definition 5.3 *A* plan *for a problem instance is a function $\pi : S \rightarrow A$ that assigns each state an action.*

A plan is executed in the obvious way: when the current state is $s \in S$, then execute $\pi(s)$ to reach a new current state, and so on. Plan execution does not terminate.

5.3 Algorithms for finding finite horizon plans

Conceptually the simplest probabilistic planning is when plan executions are restricted to have a finite horizon of length N. We briefly describe this problem to illustrate the techniques that are used in connection with the infinite horizon planning problems.

The optimum values $v_i(s)$ that can be obtained in state $s \in S$ at time point $i \in \{1, \ldots, N\}$ fulfill the following equations.

$$v_N(s) = \max_{a \in A(s)} R(s, a)$$

$$v_i(s) = \max_{a\in A(s)} \left(R(s,a) + \sum_{s'\in S} p(s'|s,a)v_{i+1}(s') \right), \text{ for } i \in \{1,\dots,N-1\}$$

The value at the last stage N is simply the best immediate reward that can be obtained, and values of states for the other stages are obtained in terms of the values of states for the later stages.

These equations also directly yield an algorithm for computing the optimal values and optimal plans: first compute v_N, then v_{N-1}, v_{N-2} and so on, until v_1 is obtained. The action to be taken in state $s \in S$ at time point i is $\pi(s,i)$ defined by

$$\begin{aligned}\pi(s,N) &= \arg\max_{a\in A(s)} R(s,a) \\ \pi(s,i) &= \arg\max_{a\in A(s)} \left(R(s,a) + \sum_{s'\in S} p(s'|s,a)v_{i+1}(s') \right), \text{ for } i \in \{1,\dots,N-1\}\end{aligned}$$

5.4 Algorithms for finding plans under discounted rewards

The value $v(s)$ of a state $s \in S$ is the discounted sum of the expected rewards that can be obtained by choosing the best possible action in s and assuming that the best possible actions are also chosen in all the possible successor states. The following equations, one for each state $s \in S$, characterize the relations between the values of states of a stochastic transition system under an optimal plan and geometrically discounted rewards with discount constant λ.

$$v(s) = \max_{a\in A(s)} \left(R(s,a) + \sum_{s'\in S} \lambda p(s'|s,a)v(s') \right) \tag{5.1}$$

These equations are called the optimality equations or the Bellman equations, and they are the basis of the most important algorithms for finding optimal plans for probabilistic planning problems with full observability.

5.4.1 Evaluating the value of a given plan

Given a plan π its value under discounted rewards with discount constant λ satisfies the following equation for every $s \in S$.

$$v(s) = R(s,\pi(s)) + \sum_{s'\in S} \lambda p(s'|s,\pi(s))v(s') \tag{5.2}$$

This yields a system of linear equation with $|S|$ equations and unknowns. The solution of these equations yields the value of the plan in each state.

5.4.2 Value iteration

The value iteration algorithm finds an approximation of the value of the optimal λ-discounted plan within a constant ϵ, and a plan with at least this value.

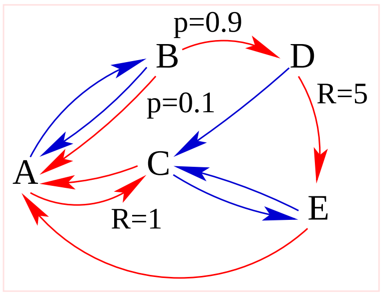


Figure 5.1: A stochastic transition system

1. $n := 0$
2. Assign (arbitrary) initial values to $v^0(s)$ for all $s \in S$.
3. For each $s \in S$, assign

$$v^{n+1}(s) := \max_{a \in A(s)} \left(R(s,a) + \sum_{s' \in S} \lambda p(s'|s,a) v^n(s') \right)$$

   If $|v^{n+1}(s) - v^n(s)| < \frac{\epsilon(1-\lambda)}{2\lambda}$ for all $s \in S$ then go to step 4.

   Otherwise, set $n := n + 1$ and go to step 3.

4. Assign

$$\pi(s) := \arg\max_{a \in A(s)} \left(R(s,a) + \sum_{s' \in S} \lambda p(s'|s,a) v^{n+1}(s') \right)$$

Theorem 5.4 *Let v_π be the value function of the plan produced by the value iteration algorithm, and let v^* be the value function of an optimal plan. Then $|v^*(s) - v_\pi(s)| \leq \epsilon$ for all $s \in S$.*

Notice that unlike in partially observable planning problems, under full observability there is never a trade-off between the values of two states: if the optimal value for state s_1 is r_1 and the optimal value for state s_2 is r_2, then there is one plan that achieves these both.

Example 5.5 Consider the stochastic transition system in Figure 5.1. Only one of the actions is nondeterministic and only in sate B, and all the other actions and states have zero reward except one of the actions in states A and D, with rewards 1 and 5, respectively. ■

5.4.3 Policy iteration

The second, also rather widely used algorithm for finding plans, is policy iteration[1]. It is slightly more complicated to implement than value iteration, but it typically converges after a smaller number of iterations, and it is guaranteed to produce an optimal plan.

The idea is to start with an arbitrary plan (assignment of actions to states), compute its value, and repeatedly choose for every state an action that is better than its old action.

[1]In connection with Markov decision processes the word *policy* is typically used instead of the word *plan*.

1. Assign $n := 0$.
2. Let π^0 be any mapping from states to actions.
3. Compute the values $v^n(s)$ of all $s \in S$ under π^n.
4. Let $\pi^{n+1}(s) = \arg\max_{a \in A(s)} \big(R(s,a) + \sum_{s' \in S} \lambda p(s'|s,a) v^n(s')\big)$.
5. Assign $n := n + 1$.
6. If $n = 1$ or $v^n \neq v^{n-1}$ then go to 3.

Theorem 5.6 *The policy iteration algorithm terminates after a finite number of steps and returns an optimal plan.*

Proof: Outline: There is only a finite number of different plans, and at each step the new plan assigns at least as high a value to each state as the old plan. □

It can be shown that the convergence rate of policy iteration is always at least as fast as that of value iteration [Puterman, 1994], that is, the number of iterations needed for finding an ϵ-optimal plan for policy iteration is never higher than the number of iterations needed by value iteration.

In practise policy iteration often finds an optimal plan after just a few iterations. However, the amount of computation in one round of policy iteration is substantially higher than in value iteration, and value iteration is often considered more practical.

5.4.4 Implementation of the algorithms with ADDs

Similar to the techniques in Section 3.7 for deterministic planning with binary decision diagrams, also probabilistic planning algorithms can be implemented with data structures that allow the representation of much bigger states spaces than what is possible by enumerative representations.

A main difference to the non-probabilistic case (Sections 3.7, 4.3 is that for probabilistic planning propositional formulae and binary decision diagrams are not suitable for representing the probabilities of nondeterministic operators nor the probabilities of the value functions needed in the value and policy iteration algorithms. However, instead of BDDs, we can use algebraic decision diagrams (Section 2.2.3).

In Section 4.1.2 we showed how the incidence matrices expressing the transition probabilities of nondeterministic operators can be represented as BDDs, when the exact probabilities can be ignored, and it is only necessary to know whether a certain nondeterministic event is possible or not.

Next we define a similar translation from nondeterministic operators to ADDs that does represent the exact probabilities.

Now we give the translation of an effect e restricted to state variables B. This means that only state variables in B may occur in e in atomic effects (but do not have to), and the formula does not say anything about the change of state variables not in B (but may of course refer to them in antecedents of conditionals.)

The last two cases, handling nondeterministic choice and conjunction of possibly nondeterministic effects is with ADD operations of multiplying an ADD with a constant, summing ADDs, and multiplying ADDs.

$$\begin{aligned}
\mathrm{PL}_B(e) &= \bigwedge(\{((a \wedge \neg \mathit{EPC}_{\neg a}(e)) \vee \mathit{EPC}_a(e)) \leftrightarrow a' | a \in B\} \\
&\quad \text{when } e \text{ is deterministic} \\
\mathrm{PL}_B(p_1 e_1 | \cdots | p_n e_n) &= p_1 \cdot \mathrm{PL}_B(e_1) + \cdots + p_n \cdot \mathrm{PL}_B(e_n) \\
\mathrm{PL}_B(e_1 \wedge \cdots \wedge e_n) &= \mathrm{PL}_{B \backslash (B_2 \cup \cdots \cup B_n)}(e_1) \cdot \mathrm{PL}_{B_2}(e_2) \cdot \ldots \cdot \mathrm{PL}_{B_n}(e_n) \\
&\quad \text{where } B_i = \mathit{changes}(e_i) \text{ for all } i \in \{1, \ldots, n\}
\end{aligned}$$

The first part of the translation $\mathrm{PL}_B(e)$ for deterministic e is the translation of deterministic effects we presented in Section 3.5.2, but restricted to state variables in B. The result of this translation is a normal propositional formula, which can be further transformed to a BDD and an ADD with only two terminal nodes 0 and 1. The other two cases cover all nondeterministic effects in normal form.

The translation of an effect e in normal form into an ADD is $\mathrm{PL}_A(e)$ where A is the set of all state variables. Translating an operators $\langle c, e \rangle$ to an ADD representing its incidence matrix is as $c \cdot \mathrm{PL}_A(e)$, where c is the ADD representing the precondition.

Example 5.7 Consider effect $(0.2\neg A|0.8A) \wedge (0.5(B \rhd \neg B)|0.5\top)$. The two conjunct translated to functions

AA'	f_A
00	0.2
01	0.8
10	0.2
11	0.8

BB'	f_B
00	1.0
01	0.0
10	0.5
11	0.5

Notice that the sum of the probabilities of the successor states is 1.0. These functions are below depicted in the same table. Notice that the third column, with the two functions componentwise multiplied, has the property that the sum of successor states of each state is 1.0.

$ABA'B'$	f_A	f_B	$f_A \cdot f_B$
0000	0.2	1.0	0.2
0001	0.2	0.0	0.0
0010	0.8	1.0	0.8
0011	0.8	0.0	0.0
0100	0.2	0.5	0.1
0101	0.2	0.5	0.1
0110	0.8	0.5	0.4
0111	0.8	0.5	0.4
1000	0.2	1.0	0.2
1001	0.2	0.0	0.0
1010	0.8	1.0	0.8
1011	0.8	0.0	0.0
1100	0.2	0.5	0.1
1101	0.2	0.5	0.1
1110	0.8	0.5	0.4
1111	0.8	0.5	0.4

■

We represent the rewards produced by operator $o = \langle c, e\rangle \in O$ in different states compactly as a list $R(o) = \{\langle\phi_1, r_1\rangle, \ldots, \langle\phi_n, r_n\rangle\}$ of pairs $\langle\phi, r\rangle$, meaning that when o is applied in a state satisfying ϕ the reward r is obtained. In any state only one of the formulae ϕ_i may be true, that is $\phi_i \models \neg\phi_j$ for all $\{i, j\} \subseteq \{1, \ldots, n\}$ such that $i \neq j$. If none of the formula is true in a given state, then the reward is zero. Hence R_o is simply a mapping from states to a real numbers.

The reward functions $R(o)$ can be easily translated to ADDs. First construct the BDDs for $\phi_1, \ldots, \phi_n$ and then multiply them with the respective rewards as

$$R_o = r_1 \cdot \phi_1 + \cdots + r_n \cdot \phi_n - \infty \cdot \neg c.$$

The summand $\infty \cdot \neg c$ handles the case in which the precondition of the operator is not satisfied: application yields immediate reward minus infinity. This prevent using the operator in any state.

Similarly, the probability distribution on possible initial states can be represented as $I = \{\langle\phi_1, p_1\rangle, \ldots, \langle\phi_n, p_n\rangle\}$ and translated to an ADD.

Now the value iteration algorithm can be rephrased in terms of ADD operations as follows.

1. Assign $n := 0$ and let v^n be an ADD that is constant 0.
2. $$v^{n+1} := \max_{\langle c,e\rangle = o \in O} \left(R_o + \lambda \cdot \exists A'.(T_o \cdot (v^n[A'/A])\right) \text{ for every } s \in S$$

   If all terminal nodes of ADD $|v^{n+1} - v^n|$ are $< \frac{\epsilon(1-\lambda)}{2\lambda}$ then stop.

   Otherwise, set $n := n + 1$ and repeat step 2.

5.5 Probabilistic planning with partial observability

5.5.1 Problem definition

5.5.2 Value iteration

Value of a plan in a state

Let $\langle C_1, \ldots, C_n\rangle$ be the partition of the state space S to the observational classes. Here $n \geq 1$.

The value of finite plans π for a state $s \in S$ is defined recursively as follows. Here $()$ is the empty plan.

$$\begin{aligned} v_{(),s} &= 0 \\ v_{(a,\pi_1,\ldots,\pi_n),s} &= \begin{cases} -\infty \text{ if action } a \text{ is not applicable in } s \\ R(s,a) + \lambda(\sum_{s' \in C_1} p(s'|s,a) v_{\pi_1,s'} + \cdots + \sum_{s' \in C_n} p(s'|s,a) v_{\pi_n,s'}) \end{cases} \end{aligned}$$

Given a belief state B and the values $v_{\pi,s_1}, \ldots, v_{\pi,s_m}$ of a plan π for all states $S \in \{s_1, \ldots, s_m\}$, the value of π for B is simply $\sum_{s \in S} v_{\pi,s} B(s)$.

Eliminating dominated plans

The test whether plan π is for at least one belief state strictly better than any other plan in $\Pi = \{\pi_1, \ldots, \pi_n\}$ can be performed by linear programming.

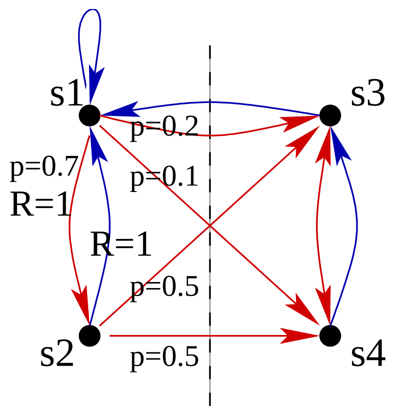


Figure 5.2: Stochastic transition system with two observational classes $\{s_1, s_2\}$ and $\{s_3, s_4\}$

The variables in the LP are d and p_s for every $s \in S$, and the expression to be maximized is the value of d. The constants $v_{\pi,s}$ are values of plans π in states $s \in S$.

$$\begin{array}{rcl}\sum_{s\in S} p_s v_{\pi,s} & \geq & \sum_{s\in S} p_s v_{\pi',s} + d \text{ for all } \pi' \in \Pi\backslash\{\pi\} \\ \sum_{s\in S} p_s & = & 1 \\ p_s & \geq & 0 \text{ for all } s \in S\end{array}$$

The number of equations in the LP is $|\Pi| + |S|$ and the number of unknowns is $|S| + 1$. If the maximum value of d is > 0, then there is a belief state in which the value of π is higher than the value of any other plan. This belief state is expressed by the values of the variables $p_s, s \in S$.

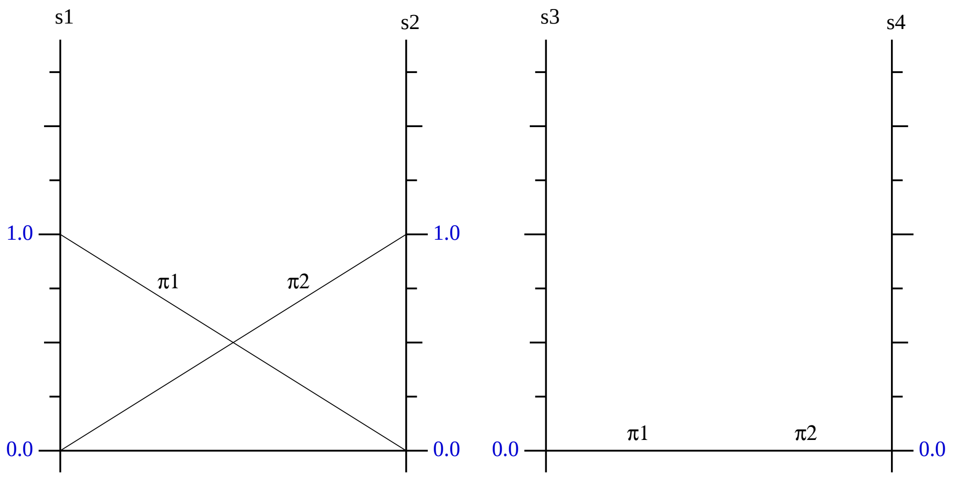
The main procedure of the algorithm

1. $i := 0$
2. $\Pi_0 := \{()\}$
3. $i := i + 1$
4. $\Pi_i := \{(a, \pi_1, \dots, \pi_n) | a \in A, \{\pi_1, \dots, \pi_n\} \subseteq \Pi_{i-1}\}$
5. Evaluate the values of plans in Π_i in all states.
6. As long as there is $\pi \in \Pi_i$ that is dominated by $\Pi_i\backslash\{\pi\}$, set $\Pi_i := \Pi_i\backslash\{\pi\}$.
7. If the difference between value functions represented by Π_i and Π_{i-1} is $> \epsilon$ for some belief state, go to 3.

Example 5.8 Consider the Now we run the value iteration algorithm for partially observable probabilistic planning problems. We use the discounting constant $\lambda = 0.5$.

Plans of depth 1 with the corresponding value vectors for all states $S = \{s_1, s_2, s_3, s_4\}$ are the following.

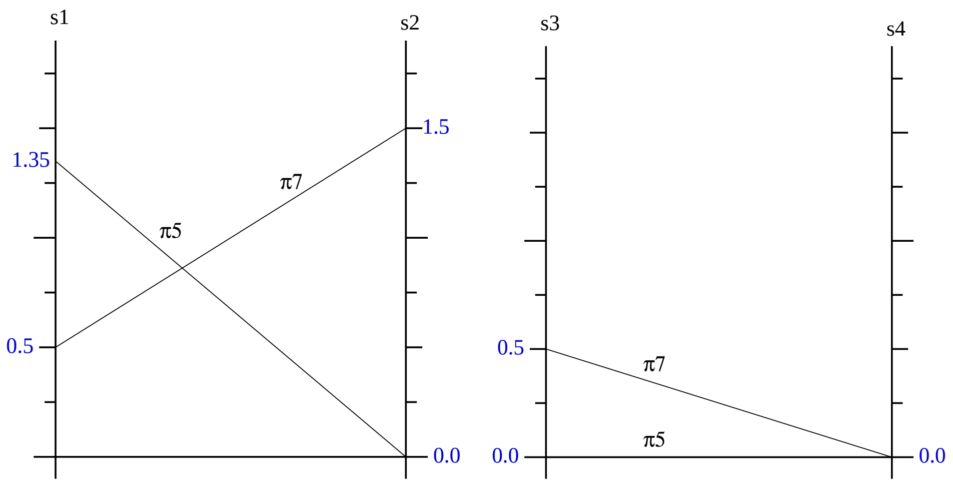
$$\begin{array}{rclrcl}\pi_1 & = & (\text{RED}, (), ()) & v(\pi_1) & = & \langle \mathbf{1.0}, \mathbf{0.0}, \mathbf{0.0}, \mathbf{0.0} \rangle \\ \pi_2 & = & (\text{BLUE}, (), ()) & v(\pi_2) & = & \langle \mathbf{0.0}, \mathbf{1.0}, \mathbf{0.0}, \mathbf{0.0} \rangle\end{array}$$



Plans of depth 2 and the corresponding value vectors are the following.

$$\begin{array}{rcl rcl}
\pi_3 & = & (\text{RED}, \pi_1, \pi_1) & v(\pi_3) & = & \langle 1.0, 0.0, 0.0, 0.0\rangle \\
\pi_4 & = & (\text{RED}, \pi_1, \pi_2) & v(\pi_4) & = & \langle 1.0, 0.0, 0.0, 0.0\rangle \\
\pi_5 & = & (\text{RED}, \pi_2, \pi_1) & v(\pi_5) & = & \langle \mathbf{1.35}, \mathbf{0.0}, \mathbf{0.0}, \mathbf{0.0}\rangle \\
\pi_6 & = & (\text{RED}, \pi_2, \pi_2) & v(\pi_6) & = & \langle 1.35, 0.0, 0.0, 0.0\rangle \\
\pi_7 & = & (\text{BLUE}, \pi_1, \pi_1) & v(\pi_7) & = & \langle \mathbf{0.5}, \mathbf{1.5}, \mathbf{0.5}, \mathbf{0.0}\rangle \\
\pi_8 & = & (\text{BLUE}, \pi_1, \pi_2) & v(\pi_8) & = & \langle 0.5, 1.5, 0.5, 0.0\rangle \\
\pi_9 & = & (\text{BLUE}, \pi_2, \pi_1) & v(\pi_9) & = & \langle 0.0, 1.0, 0.0, 0.0\rangle \\
\pi_{10} & = & (\text{BLUE}, \pi_2, \pi_2) & v(\pi_{10}) & = & \langle 0.0, 1.0, 0.0, 0.0\rangle
\end{array}$$

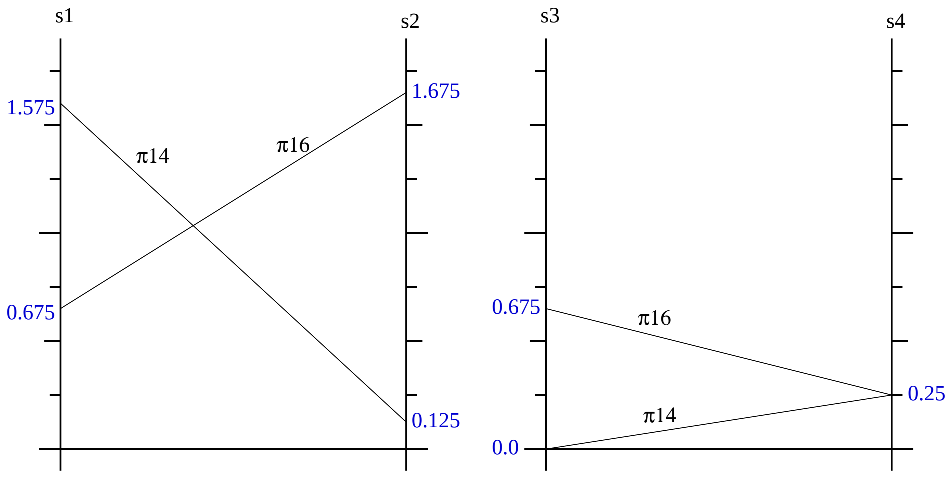
The graphical representation of these vectors is as follows.



Because s_1 and s_2 are indistinguishable, but distinguishable from s_3 and s_4, the associated value functions can be depicted in two diagrams, each depicting probability distributions on a set of states that are indistinguishable from each other.

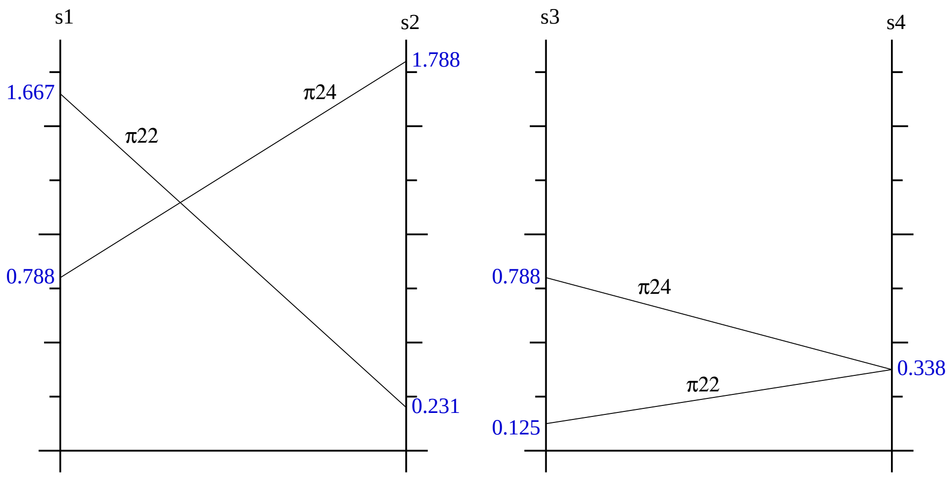
When enumerating plans of depth $i + 1$, it suffices to use as subplans only those plans of depth i that are the best plans for at least one belief state. Hence from the depth 2 plans we can ignore all but π_5 and π_7. Notice that in this example, we accidentally can recognize those plans that are better anywhere from the fact that they are strictly worse on all states, and all the remaining plans are strictly better in at least one state. Plans of depth 3 and the corresponding value vectors are the following.

$$\begin{array}{llllll}
\pi_{11} & = & (\text{RED}, \pi_5, \pi_5) & v(\pi_{11}) & = & \langle 1.0, 0.0, 0.0, 0.0\rangle \\
\pi_{12} & = & (\text{RED}, \pi_5, \pi_7) & v(\pi_{12}) & = & \langle 1.05, 0.125, 0.0, 0.25\rangle \\
\pi_{13} & = & (\text{RED}, \pi_7, \pi_5) & v(\pi_{13}) & = & \langle 1.525, 0.0, 0.0, 0.0\rangle \\
\pi_{14} & = & (\text{RED}, \pi_7, \pi_7) & v(\pi_{14}) & = & \langle \mathbf{1.575}, \mathbf{0.125}, \mathbf{0.0}, \mathbf{0.25}\rangle \\
\pi_{15} & = & (\text{BLUE}, \pi_5, \pi_5) & v(\pi_{15}) & = & \langle 0.675, 1.675, 0.675, 0.0\rangle \\
\pi_{16} & = & (\text{BLUE}, \pi_5, \pi_7) & v(\pi_{16}) & = & \langle \mathbf{0.675}, \mathbf{1.675}, \mathbf{0.675}, \mathbf{0.25}\rangle \\
\pi_{17} & = & (\text{BLUE}, \pi_7, \pi_5) & v(\pi_{17}) & = & \langle 0.25, 1.25, 0.25, 0.0\rangle \\
\pi_{18} & = & (\text{BLUE}, \pi_7, \pi_7) & v(\pi_{18}) & = & \langle 0.25, 1.25, 0.25, 0.25\rangle
\end{array}$$

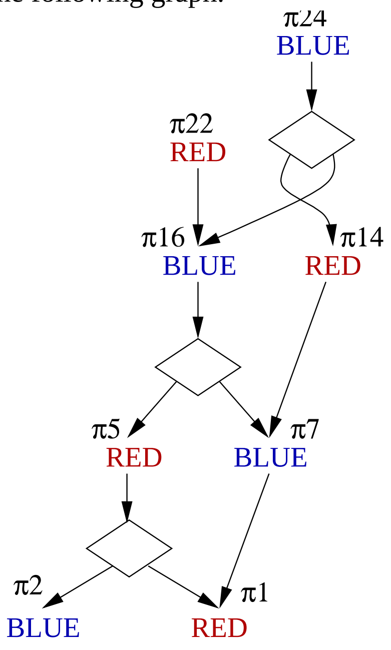


Plans of depth 4 and the corresponding value vectors are the following.

$$\begin{array}{llllll}
\pi_{19} & = & (\text{RED}, \pi_{14}, \pi_{14}) & v(\pi_{19}) & = & \langle 1.05625, 0.0625, 0.125, 0.0\rangle \\
\pi_{20} & = & (\text{RED}, \pi_{14}, \pi_{16}) & v(\pi_{20}) & = & \langle 1.12375, 0.23125, 0.125, 0.3375\rangle \\
\pi_{21} & = & (\text{RED}, \pi_{16}, \pi_{14}) & v(\pi_{21}) & = & \langle 1.59875, 0.0625, 0.125, 0.0\rangle \\
\pi_{22} & = & (\text{RED}, \pi_{16}, \pi_{16}) & v(\pi_{22}) & = & \langle \mathbf{1.66625}, \mathbf{0.23125}, \mathbf{0.125}, \mathbf{0.3375}\rangle \\
\pi_{23} & = & (\text{BLUE}, \pi_{14}, \pi_{14}) & v(\pi_{23}) & = & \langle 0.7875, 1.7875, 0.7875, 0.0\rangle \\
\pi_{24} & = & (\text{BLUE}, \pi_{14}, \pi_{16}) & v(\pi_{24}) & = & \langle \mathbf{0.7875}, \mathbf{1.7875}, \mathbf{0.7875}, \mathbf{0.3375}\rangle \\
\pi_{25} & = & (\text{BLUE}, \pi_{16}, \pi_{14}) & v(\pi_{25}) & = & \langle 0.3375, 1.3375, 0.3375, 0.0\rangle \\
\pi_{26} & = & (\text{BLUE}, \pi_{16}, \pi_{16}) & v(\pi_{26}) & = & \langle 0.3375, 1.3375, 0.3375, 0.3375\rangle
\end{array}$$



The plan can be depicted as the following graph.



■

5.6 Literature

A comprehensive book on (fully observable) Markov decision processes has been written by Puterman [1994], and our presentation of the algorithms in Section 5.4 (5.4.2 and 5.4.3) follows that of Puterman. The book represents the traditional research on MDPs and uses exclusively enumerative representations of state spaces and transition probabilities. The book discusses all the main optimality criteria as well as algorithms for solving MDPs by iterative techniques and linear programming. There are also many other books on solving MDPs.

A planning system that implements the value iteration algorithm with ADDs is described by Hoey et al. [1999] and is shown to be capable of solving problems that could not be efficiently solved by conventional implementations of value iteration.

The best known algorithms for solving partially observable Markov decision processes were presented by Sondik and Smallwood in the early 1970's [Sondik, 1978; Smallwood and Sondik, 1973] and even today most of the work on POMDPs is based on those algorithms [Kaelbling *et al.*, 1998]. In this section we have presented the standard value iteration algorithm with the simplification that there is no sensing uncertainty, that is, for every state the same observation, dependent on the state, is always made.

The most general infinite-horizon planning problems and POMDP solution construction are undecidable [Madani *et al.*, 2003]. The complexity of probabilistic planning has been investigated for example by Mundhenk et al. [2000] and Littman [1997].

Bonet and Geffner [2000] and Hansen and Zilberstein [2001] have presented algorithms for probabilistic planning with Markov decision processes that use heuristic search.

5.7 Exercises

5.1 Prove that on each step of policy iteration the policy improves.