

CS533
Intelligent Agents and Decision Making
Homework 3, Winter 2009

1. Prove that for any STRIPS planning problem that the length of the optimal relaxed plan from any state s to the goal g is an admissible heuristic for state s . (NOTE: This does not require a complicated proof.)

Solution: Consider a STRIPS planning problem S , its corresponding relaxed planning problem R , and a plan P (need not be a solution plan). Let s' be the state reached in S after executing plan P starting at s . Likewise let s'' be the state reached in R after executing plan P starting at s . One can show by induction on the plan length that it is always the case that $s' \subseteq s''$. This is true because actions in the relaxed plans add all of the facts that “unrelaxed” actions do, but do not delete any facts.

From the above observation it follows that for any state s , if a plan P reaches a goal from s in S then it will also reach the goal from s in R . In other words, all solutions to a planning problem are solutions to its relaxed problem.

Given this observation, we know that for any state the optimal solution plan P^* from s to the goal in S is also a solution plan from s to the goal in R . This implies that the optimal plan from s to the goal in R must be no longer than P^* , showing that the length of the optimal plan in R is an admissible estimate for the length of the optimal plan in S .

2. Consider the Sussman anomaly problem in Figure 11.16 of your book. Write a definition for this problem in STRIPS (as you did in HW1). Using this definition, compute the HSP heuristics h_{\max} and h_{add} for the initial state of the problem. Show the steps of each computation.

Solution:

Below we will use T to represent the table.

The initial state is:

$\{\text{On}(B,T), \text{On}(C,A), \text{On}(A,T), \text{Clear}(B), \text{Clear}(C)\}$

The goal is:

$\{\text{On}(A,B), \text{On}(B,C), \text{On}(C,T)\}$

We will use the following operator schemas:

Move(X,Y,Z)
PRE: On(X,Y), Clear(X), Clear(Z)
ADD: On(X,Z), Clear(Y)
DEL: On(X,Y), Clear(Z)

MoveToTable(X,Y)
PRE: On(X,Y), Clear(X)
ADD: On(X,T), Clear(Y)
DEL: On(X,Y)

To compute the heuristic value $h_{\text{add}}(s)$ where s is the initial state we can run the iterative algorithm described in class and in the HSP paper. Recall that each iteration of the algorithm updates the heuristic estimate $\Delta_0(p)$ for each state fact p . Initially the estimates of facts in s are equal to 0 and all other estimates are set equal to infinity. Each iteration will consider whether it is possible to decrease the estimate by considering each action that could add the fact. The algorithm terminates when two iterations produce the same estimates. Below the first column shows all of the facts for our problem. The next three columns show the estimates of Δ_0 produced for each fact after each iteration of the algorithm. The final column gives the final estimate for each fact.

	iteration			
	0	1	2	3
On(A,B)	inf	inf	2	2
On(A,C)	inf	inf	2	2
On(A,T)	0	0	0	0
On(B,A)	inf	inf	2	2
On(B,C)	inf	1	1	1
On(B,T)	0	0	0	0
On(C,A)	0	0	0	0
On(C,B)	inf	1	1	1
On(C,T)	inf	1	1	1
Clear(A)	inf	1	1	1
Clear(B)	0	0	0	0
Clear(C)	0	0	0	0

Given these estimates $h_{\text{max}}(s) = \sum_{p \in G} \Delta_0(p)$ where G is the goal. In our case this gives us,

$$h_{\text{add}}(s) = \Delta_0(\text{On}(A,B)) + \Delta_0(\text{On}(B,C)) + \Delta_0(\text{On}(C,T)) = 4$$

This is clearly an overestimate, since there is a plan that takes only three steps. This shows that h_{add} is not admissible.

To compute $h_{\text{max}}(s)$ we would run an algorithm like the one above, but replace the summations with maximizations. It turns out that if we do this we will get the exact same estimates of Δ_0 as above. However, the final heuristic value will now be,

$$h_{\text{max}}(s) = \max\{\Delta_0(\text{On}(A,B)), \Delta_0(\text{On}(B,C)), \Delta_0(\text{On}(C,T))\} = 2$$

This is an underestimate of the distance to goal, which should be the case since h_{max} is an admissible heuristic.

3. Exercise 17.3 For this you can assume that the search space is such that each operator has unit cost. Note that a STRIPS planning problem can be cast as this type of finite search problem.

Solution: A finite search problem (see Chapter 3) is defined by an initial state s_0 , a successor function $S(s, a)$ that returns a successor state for action a in state s . An optimal solution is a shortest path from s_0 to any goal state.

To construct the corresponding MDP, define $R(s) = -1$ unless s is a goal state, in which case $R(s) = 0$. Define $T(s, a, s') = 1$ if $S(s, a) = s'$ and $T(s, a, s') = 0$ otherwise. Also let the discount factor $\gamma = 1$. An optimal solution to this MDP is a policy that follows the lead-cost path from each state to its nearest goal state.

4. Exercise 17.4 b and first question of part c (NOTES: By terminal state the book means a state with zero reward that can only transition to itself. By undiscounted MDP the book means an infinite horizon MDP where the discount factor is set equal to 1.)

Solution:

Part B)

The application of policy iteration proceeds in alternating steps of value determination and policy update.

Initialization $\pi = (b; b)$. We evaluate this initial policy by writing out the system of linear equations that V_π must satisfy. We will denote $V_\pi(s_i)$ by V_i .

$$\begin{aligned} V_1 &= -1 + 0.1V_3 + 0.9V_1 \\ V_2 &= -2 + 0.1V_3 + 0.9V_2 \\ V_3 &= 0 \end{aligned}$$

Solving this system yields $V_1 = -10$ and $V_2 = -20$.

Now using the value function for π we define a new improved policy in the policy update step. Recall that in each state we will consider each action and then select the best one. In s_1 for action a we get, $\sum_j T(s_1, a, s_j)V_j = 0.8 \cdot -20 + 0.2 \cdot -10 = -18$ and for action b get $\sum_j T(s_1, b, s_j)V_j = 0.1 \cdot 0 + 0.9 \cdot -10 = -9$, so we will still prefer action b . In the same way for s_2 we get a value of -12 for action a and -18 for action b . So now action a is preferred for s_2 rather than action b .

Our new improved policy is $\pi = (b; a)$. The equations for value determination become,

$$\begin{aligned} V_1 &= -1 + 0.1V_3 + 0.9V_1 \\ V_2 &= -2 + 0.8V_1 + 0.2V_2 \\ V_3 &= 0 \end{aligned}$$

which yields $V_1 = -10$ and $V_2 = -15$. Notice that the value of s_2 has improved. Going through the same steps as above for the policy update we get that in s_1 actions a and b have values -14 and -9 respectively and in s_2 actions a and b have values -11 and -13.5 respectively. Thus, we will prefer b in s_1 and a in s_2 . Since this is the same policy that we started with we are guaranteed that $\pi = (b; a)$ is an optimal policy.

Part C)

An initial policy with action a in both states intuitively has a value of negative infinity, since on each step a negative reward will be received. If we attempt to evaluate the policy $\pi = (a; a)$ then we get the equations,

$$\begin{aligned} V_1 &= -1 + 0.2V_1 + 0.8V_2 \\ V_2 &= -2 + 0.8V_1 + 0.2V_2 \\ V_3 &= 0 \end{aligned}$$

and the first two equations are inconsistent. If you try to use Matlab to solve the system then it will report the system is ill conditioned. If you used an iterative procedure to solve the system then the values would tend to move toward negative infinity. Discounting leads to well-defined solutions by bounding the penalty an agent can incur at either state. However, the choice of the discount factor will effect the policy that results. For small γ the cost incurred in the distant future plays a negligible role in the value computation, because γ^n is near zero. As a result, an agent could choose action b in s_2 because the discounted short-term cost of remaining in the non-terminal states (states 1 and 2) outweighs the discounted long-term cost of action b failing repeatedly and leaving the agent in s_2 .

5. Exercise 17.5 (a terminal state is a state with zero reward that can only transition to itself)

(a) The key here is to get the max and summations in the right place. For $R(s,a)$ we have,

$$V(s) = \max_a [R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s')]$$

and for $R(s, a, s')$ we have,

$$V(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a) + \gamma V(s')]$$

(b) There are a variety of solutions here. One is to create a new “pre-state” $\text{pre}(s,a,s')$ for every s, a, s' , such that executing a in s leads not to s' but to $\text{pre}(s,a,s')$. In this state is encoded the fact that the agent came from s and did a to get there. From the pre-state there is just one action b that always leads to s' . Let the new MDP have transition T' , reward R' , and discount factor γ' , as defined below.

$$\begin{aligned} T'(s, a, \text{pre}(s, a, s')) &= T(s, a, s') \\ T'(\text{pre}(s, a, s'), b, s') &= 1 \\ R'(s, a) &= 0 \\ R'(\text{pre}(s, a, s'), b) &= \gamma^{-0.5} R(s, a, s') \\ \gamma' &= \gamma^{0.5} \end{aligned}$$

This MDP has the property that for any policy, the value of that policy in any state s of the original MDP is the same as the value in state s of the new MDP.

Note that the new discount factor is necessary to account for the fact that for each step in the original MDP, there are two steps in the new MDP. In particular, consider an action-state sequence in the original MDP that would have produced the reward sequence r_1, r_2, r_3 giving a discounted value of $r_1 + \gamma r_2 + \gamma^2 r_3$. In our new MDP with the inclusion of the “post” states, this reward sequence would translate to

$$0, \gamma^{-0.5} r_1, 0, \gamma^{-0.5} r_2, 0, \gamma^{-0.5} r_3$$

and the corresponding value would be

$$0 + \gamma' \gamma^{-0.5} r_1 + \gamma'^2 0 + \gamma'^3 \gamma^{-0.5} r_2 + \gamma'^4 0 + \gamma'^5 \gamma^{-0.5} r_3$$

which is equal to,

$$\gamma^{0.5} \gamma^{-0.5} r_1 + \gamma^{0.5 \cdot 3} \gamma^{-0.5} r_2 + \gamma^{0.5 \cdot 5} \gamma^{-0.5} r_3 = r_1 + \gamma r_2 + \gamma^2 r_3$$

as desired.

- (c) We can use the same idea as above. For every s, a we will create a new state $\text{post}(s, a)$, such that

$$\begin{aligned} T'(s, a, \text{post}(s, a)) &= 1 \\ T'(\text{post}(s, a), b, s') &= T(s, a, s') \\ R'(s, a) &= 0 \\ R'(\text{post}(s, a), b) &= \gamma^{-0.5} R(s, b) \\ \gamma' &= \gamma^{0.5} \end{aligned}$$

6. **(Policy Evaluation.)** Given a policy π , let V_π be the infinite-horizon, discounted value function (as defined in class), which we know satisfies the following equation at all states s ,

$$V_\pi(s) = R(s) + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot V_\pi(s') \quad (1)$$

We can compute V_π by solving the above system of linear equations. However, there is also an iterative technique for computing V_π that is often more efficient. Consider the following value-function operator T_π ,

$$T_\pi[V](s) = R(s) + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot V(s')$$

note that $T_\pi[V]$ is simply a value function and $T_\pi[V](s)$ gives the value of state s . As described in your book in the discussion of modified policy iteration, this operator can be used to iteratively compute a sequence of value functions V^k that converge to V_π as follows:

$$\begin{aligned} V^0(s) &= 0, \text{ for all } s \\ V^k &= T_\pi[V^{k-1}] \end{aligned}$$

Use the following steps to prove that the sequence does converge to the correct value function.

- (a) Show that T_π is a contraction operator with respect to the max-norm. That is show that for any value functions V and V' ,

$$\|T_\pi[V] - T_\pi[V']\| \leq \gamma \|V - V'\|$$

We will show that for every state s ,

$$|T_\pi[V](s) - T_\pi[V'](s)| \leq \gamma \|V - V'\|$$

which by the definition of the max-norm implies our desired result. This can be shown using the following steps.

$$\begin{aligned} |T_\pi[V](s) - T_\pi[V'](s)| &= |R(s) + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot V(s') - R(s) + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot V'(s')| \\ &= |\gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot (V(s') - V'(s'))| \\ &\leq \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot |V(s') - V'(s')| \\ &\leq \gamma \sum_{s'} \Pr(s'|s, \pi(s)) \cdot \|V - V'\| \\ &= \gamma \|V - V'\| \sum_{s'} \Pr(s'|s, \pi(s)) \\ &= \gamma \|V - V'\| \end{aligned}$$

The first and second line follow from the definition of T_π and simple algebra. The third line follows because for any sequence of numbers, the absolute value of the sum is never greater than the sum of the absolute values of the numbers. The fourth line follows from the definition of the max-norm. The remaining lines are straightforward.

- (b) Use this fact to prove that $\lim_{k \rightarrow \infty} V^k = V_\pi$. You may use equation 1 if desired.

From equation 11 we see that V_π is a fixed point of T_π —i.e. $T_\pi[V_\pi] = V_\pi$. We use this fact to prove that for any $k \geq 0$, $\|V^k - V_\pi\| \leq \gamma^k \|V^0 - V_\pi\|$. We prove this by induction on k .

The base case of $k = 0$ is trivial. We show the statement is true for $k + 1$ as follows,

$$\begin{aligned} \|V^{k+1} - V_\pi\| &= \|T_\pi[V^k] - T_\pi[V_\pi]\| \\ &\leq \gamma \|V^k - V_\pi\| \\ &\leq \gamma \gamma^k \|V^0 - V_\pi\| \\ &= \gamma^{k+1} \|V^0 - V_\pi\| \end{aligned}$$

The first line follows from our definition of V^k and the fixed point property of V_π . The second line follows from the contraction property. The third line follows from the inductive hypothesis.

Using this property we get that,

$$\lim_{k \rightarrow \infty} \|V^k - V_\pi\| \leq \lim_{k \rightarrow \infty} \gamma^k \|V^0 - V_\pi\| = 0$$

Since the max-norm must be positive we know that $\lim_{k \rightarrow \infty} \|V^k - V_\pi\| = 0$, which implies that $\lim_{k \rightarrow \infty} V^k = V_\pi$.

- (c) Does the sequence still converge to V_π if we initialize V^0 to random values? Explain.
Yes. As long as V_0 is bounded the above proof still works.
- (d) What value of k is sufficient so that $\|V^k - V_\pi\| \leq \epsilon$? Explain.

This follows the same argument as used to bound k for value iteration. Above we showed that $\|V^k - V_\pi\| \leq \gamma^k \|V^0 - V_\pi\|$. Assume that $V^0 = 0$ then we know that $\|V^0 - V_\pi\| \leq \frac{R_{\max}}{1-\gamma}$, which shows that $\|V^k - V_\pi\| \leq \gamma^k \frac{R_{\max}}{1-\gamma}$. So it suffices to find a lower bound on k so that $\gamma^k \frac{R_{\max}}{1-\gamma} < \epsilon$. Taking the log of both sides and rearranging (noting that $\log(1-\gamma) < 0$), we get that,

$$k > \frac{\log \frac{\epsilon(1-\gamma)}{R_{\max}}}{\log \gamma}$$

is sufficient to ensure an error no more than ϵ .