AI Planning for Autonomy

Solution Problem Set I: Blind Search

1.

Consider a set of cities V to visit in any order, a starting city location v_{start}, and a set of edges E specifying if there's an edge from two cities $\langle v, v' \rangle$:

$$
\begin{aligned}
S &= \{\langle current_v, V' \rangle | \, current_v \in V \wedge V' \subseteq V\} \\
S_0 &= \langle v_{start}, \{v_{start}\} \rangle \\
A(\langle current_v, V' \rangle) &= \{\langle current_v, v' \rangle | \, \langle current_v, v' \rangle \in E\} \\
f(\langle current_v, V' \rangle, \langle current_v, v' \rangle) &= \langle v', V' \cup v' \rangle \\
c(a, s) &= cost(edge) \\
S_G &= \{\langle current_v, V \rangle\}
\end{aligned}
$$

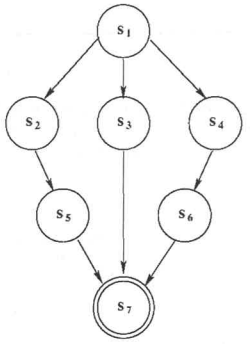
2.

- Which is the solution found by each algorithm?
  - BRFS,ID: $s_1 \rightarrow s_3 \rightarrow s_7$
  - DFS: Depends, could be any of the 3 solutions.
- Which is the optimal solution?
  - $s_1 \rightarrow s_4 \rightarrow s_6 \rightarrow s_7$
- Explain under which conditions the algorithms guarantee optimality?
  - BRFS and ID will be optimal if the costs are uniform. For example: all costs are 1.
- Adapt any of the previous algorithms to account for $g(n)$. Explain properties: optimality,complete,sound.
  - Dijkstra, also known as Uniform search will be optimal and complete. It's Like BRFS but expanding first the node with lowest accumulated cost, instead of the lowest length, where length is defined over the number of traversed arcs. By construction Dijktra is sound, meaning that any returned path is a path that exists in the graph.

AI Planning for Autonomy

Problem Set II: Heuristic Search Continued

1. Consider the following state space S, where $s_0 = s_1$ and $S_G = \{s_7\}$



where actions changing a state s into another state s' are given by the edges. The cost to transition from state s to s' is given by the following table:

s	s'	$c(s,s')$	s	s'	$c(s,s')$
s_1	s_2	2	s_3	s_7	10
s_1	s_3	2	s_4	s_6	1
s_1	s_4	1	s_5	s_7	3
s_2	s_5	2	s_6	s_7	4

and heuristic estimates for each state:

s	$h_1(s)$	$h_2(s)$	$h_3(s)$
s_1	4	6	6
s_2	3	5	1
s_3	5	10	1
s_4	3	5	5
s_5	2	3	3
s_6	2	4	4
s_7	0	0	0

- Which heuristics are admissible?
  All 3
- Which are consistent?
  h_1 and h_2 are consistent, h_3 is not because $h_3(s_7) < h_3(s_4) + c(s_1, s_2)$
- Does any heuristic *dominate* any other?
  $h_2 = h^*$ and therefore dominates all other admissible heuristics. $h_1(s_1) < h_3(s_1)$ and $h_3(s_2) < h_1(s_2)$ therefore neither of h_1 and h_3 dominate each other

Describe the execution of one of the following algorithms in this problem using one of the heuristics above. Fill in a table like the one below, showing the contents of the OPEN and CLOSED lists at the end of each iteration.

Choose one of: A*, WA* ($w = 5$), or Greedy Best-First Search.

Using A and h_2, labeling each node n_i whose parent is n_p as $n_i = \langle s, g(n_i) + h(s), g(n_i), n_p \rangle$*

	Iteration 1	Iteration 2	Iteration 3	Iteration 4
OPEN	$n_1 = \langle s_1, 6, 0, nil \rangle$ *	$n_2 = \langle s_2, 7, 2, n_1 \rangle$ $n_3 = \langle s_3, 12, 2, n_1 \rangle$ $n_4 = \langle s_4, 6, 1, n_1 \rangle$ *	n_2 n_3 $n_5 = \langle s_6, 6, 2, n_4 \rangle$ *	n_2 n_3 $n_6 = \langle s_7, 6, 6, n_5 \rangle$ *
CLOSED		n_1	n_1 n_4	n_1 n_4 n_5

- Which is the path returned as a solution?
  $s_1 \to s_4 \to s_6 \to s_7$
- Is this the optimal plan? Has the algorithm proved this?
  Yes, since n_6 is at a goal state and has the lowest admissible cost estimate of any node in the open list in iteration 4, all other paths from any open node must be longer, given h_2 is both admissible and consistent.

2. Consider an $m \times m$ manhattan grid, and a set of coordinates G to visit in any order.
   - Formulate a state-based search problem to find a tour of all the desired points (i.e. define a state space, applicable actions, transition and cost functions).

$$
\begin{aligned}
S &= \{\langle x, y, V \rangle \mid x, y \in \{0, \ldots, m-1\} \wedge V \subseteq G\} \\
A(\langle x, y, V \rangle) &= \{(dx, dy) \mid dx, dy \in \{-1, 0, 1\} \\
&\qquad \wedge\ |dx| + |dy| = 1 \\
&\qquad \wedge\ x + dx \in \{0, \ldots, m-1\} \\
&\qquad \wedge\ y + dy \in \{0, \ldots, m-1\}\} \\
t((dx, dy), \langle x, y, V \rangle) &= \langle x + dx, y + dy, V \setminus \{(x + dx, y + dy)\} \rangle \\
c(a, s) &= 1
\end{aligned}
$$

   - What is the branching factor of the search?
     approx. 4
   - What is the size of the state space in terms of m and G.
     approx. $m^2 \cdot 2^{|G|}$
   - Define an admissible heuristic function.

$$h(\langle x, y, V \rangle) = \max_{(x_g, y_g) \in V} (|x - x_g| + |y - y_g|)$$

AI Planning for Autonomy

Solution Problem Set III: Choosing Heuristics

1.

x and y are coordinates, and v are the coordinates that remain to be visited. The rest of the questions are discussion questions for class or on LMS forum.

2.

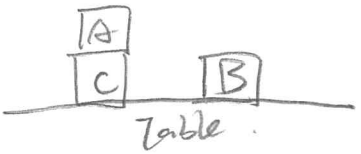
F={ at(x,y), visited(x,y) | x,y $\in \{0, \ldots, m-1\}$ },
A= { move(x,y,x',y'):

- Prec: at(x,y)
- Add: at(x',y'), visited(x',y')
- Del: at(x,y)

| for each adjacent (x,y) (x',y'), and (x',y') $\notin$W }
I = {at(0,0)}
F = {visited(x,y) | (x,y) $\in$ V}

AI Planning for Autonomy

Solution Problem Set IV: Modeling in STRIPS and PDDL



1.

For a given set of Blocks B:

F={ on(x,y), onTable(x), clear(x), holding(x), armFree | $x, y \in B$ },

A= { stack(x,y):

- Prec: holding(x), clear(y)
- Add: clear(x), on(x,y), armFree
- Del: clear(y), holding(x)

| $x, y \in B$ }

A = A∪ { unstack(x,y):

- Prec: on(x,y), clear(x), armFree
- Add: holding(x), clear(y)
- Del: clear(x), on(x,y), armFree

| $x, y \in B$ }

A= A∪ { putdown(x):

- Prec: holding(x)
- Add: clear(x), onTable(x), armFree
- Del: holding(x)

| $x \in B$ }

A = A∪ { pickup(x):

- Prec: onTable(x), clear(x), armFree
- Add: holding(x)
- Del: clear(x), onTable(x), armFree

| $x \in B$ }

I = {on(a,c), onTable(c), onTable(b), clear(a), clear(b), armFree}

G = { on(a,b), on(b,c) }

For a given set of Block B.

F = { on(x,y), onTable(x), clear(x), holding(x), armFree | x,y ∈ B }

A = { stack(x,y):
pre: holding(x), clear(y)
add: on(x,y), clear(x), armFree
delete: holdy(x), clear(y)
| x,y ∈ B }

A = A ∪ { unstack(x,y):
pre = armFree(), clear(x), on(x,y)
add: holdly(x), clear(y),
del: armFree(), clear(x), on(x,y).
| x,y ∈ B }

A = A ∪ { putup(x):
pre: armFree(), onTable(x), clear(x)
add: holdly(x)
del: armFree(), onTable(x), clear(x)
| x ∈ B }

A = A ∪ { putdown(x): ✓
pre: holding(x).
add: onTable(x), clear(x), armFree ☆
del: holdly(x). | x ∈ B }

2-3. PDDL domains have been created here: `http://editor.planning.domains/#read_session=TYXYMCQN8w`

I = { on(A,C), clear(A), onTable(C), onTable(B), clear(B), armFree() }

G = { on(A,B), on(B,C) }

AI Planning for Autonomy

Solution Problem Set V: Delete Relaxation

1.

- If computed with respect to each food it's roughly a Minimum Spanning Tree (technically a Steiner Tree, since paths can branch in non-food location, i.e. the Steiner Points)
- $h_{max} << h^+ << h^*$, $h_{max} << h^+ << h_{add}$. h^* dominates admissible heuristics, that's why it doesn't dominate h_{add}.

2.

- Compute $h^{add}(s_0)$ for this blocks-world problem. $h^{add}(s_0) = 5$. For computation, see below.
- Compute $h^{max}(s_0)$ for this blocks-world problem. $h^{max}(s_0) = 2$. For computation, see below.

I cannot instantiate on(x,y)

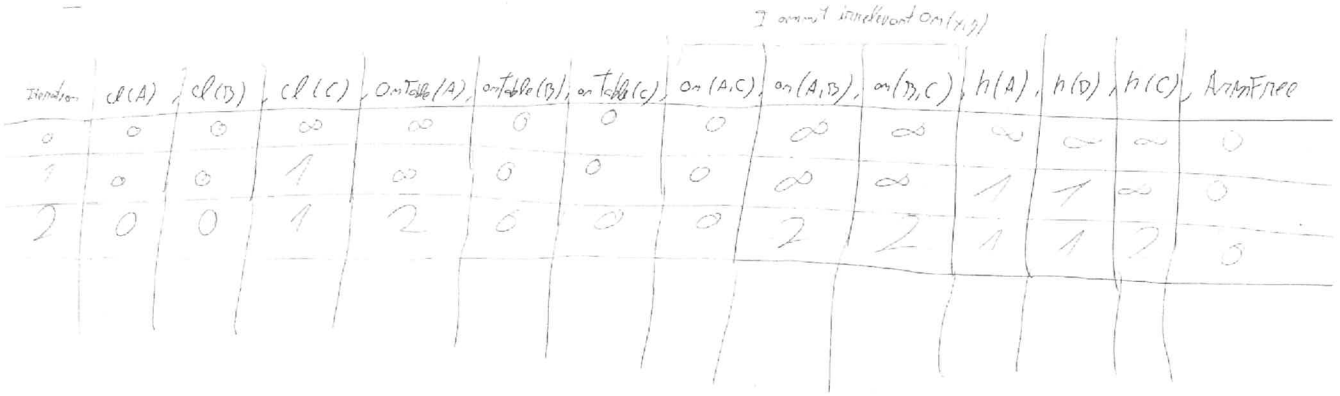
Iteration	cl(A)	cl(B)	cl(C)	onTable(A)	onTable(B)	onTable(C)	on(A,C)	on(A,B)	on(B,C)	h(A)	h(B)	h(C)	ArmFree
0	0	0	∞	∞	0	0	0	∞	∞	∞	∞	∞	0
1	0	0	1	∞	0	0	0	∞	∞	1	1	∞	0
2	0	0	1	2	0	0	0	2	2	1	1	2	0

The table for h_{add} changes only the value for $on(B, C)$ to 3, hence h value of the Goal is 5.

AI Planning for Autonomy

Solution Problem Set VI: Relaxed Plan Heuristic and Iterated Width

1. Derive best supporters function from the last row of the h^{max} table.



The table for h^{add} changes only the value for $on(B, C)$ to 3.

$h_{ff} = 4$ for both cases. Even if $h_{add}(s_0, G) \neq h_{max}(s_0, G)$, the best supporter bs function doesn't change.

$bs(on(A, B)) = Stack(A, B) \rightarrow$ need to support precs: $holding(A)$. Supported by initial state: $clear(B)$

$bs(on(B, C)) = Stack(B, C) \rightarrow$ need to support precs: $holding(A)$ and $clear(C)$

$bs(holding(A)) = Unstack(A, C) \rightarrow$ all precs supported by initial state

$bs(holding(B)) = Pickup(B) \rightarrow$ all precs supported by initial state

$bs(clear(C)) = Unstack(A, C) \rightarrow$ all precs supported by initial state.

Relaxed Plan is = { Unstack(A,C), Pick(B), stack(A,B), stack(B,C) }.

Note that even if $Unstack(A, C)$ appears twice as a selected best supporter, it is only considered once in the relaxed plan. Each time the relaxed plan $RelPlan$ is extended by the set union operator $RelPlan = RelPlan \cup bs(g)$, and this operation over sets does not create duplicates by definition.

2.

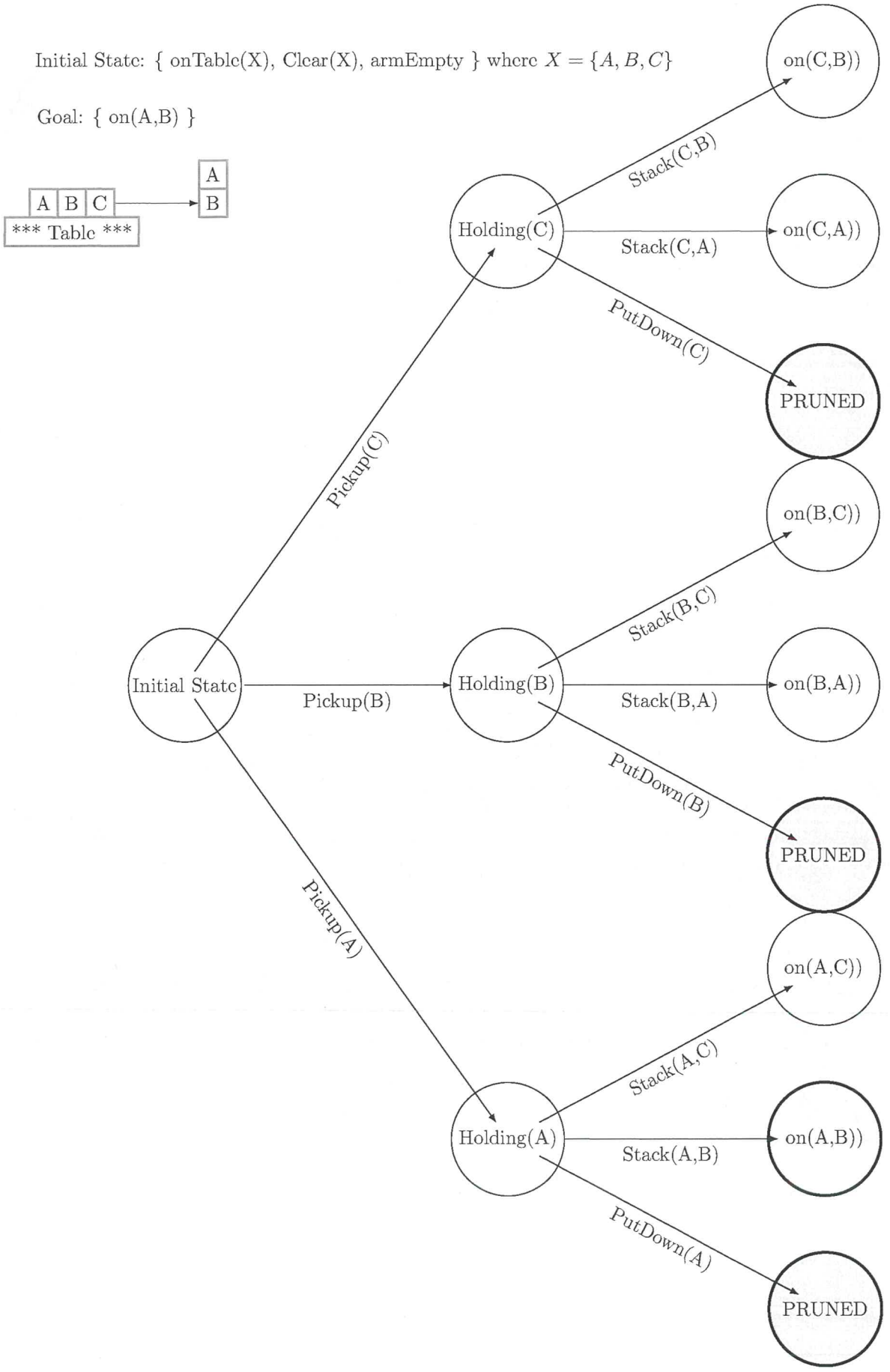


Figure 1: IW tree for question 2.a) Each node shows the atomic fluents that state makes true for the first time. States that do not add a new atom for the first time are pruned. A table can be induced from this tree

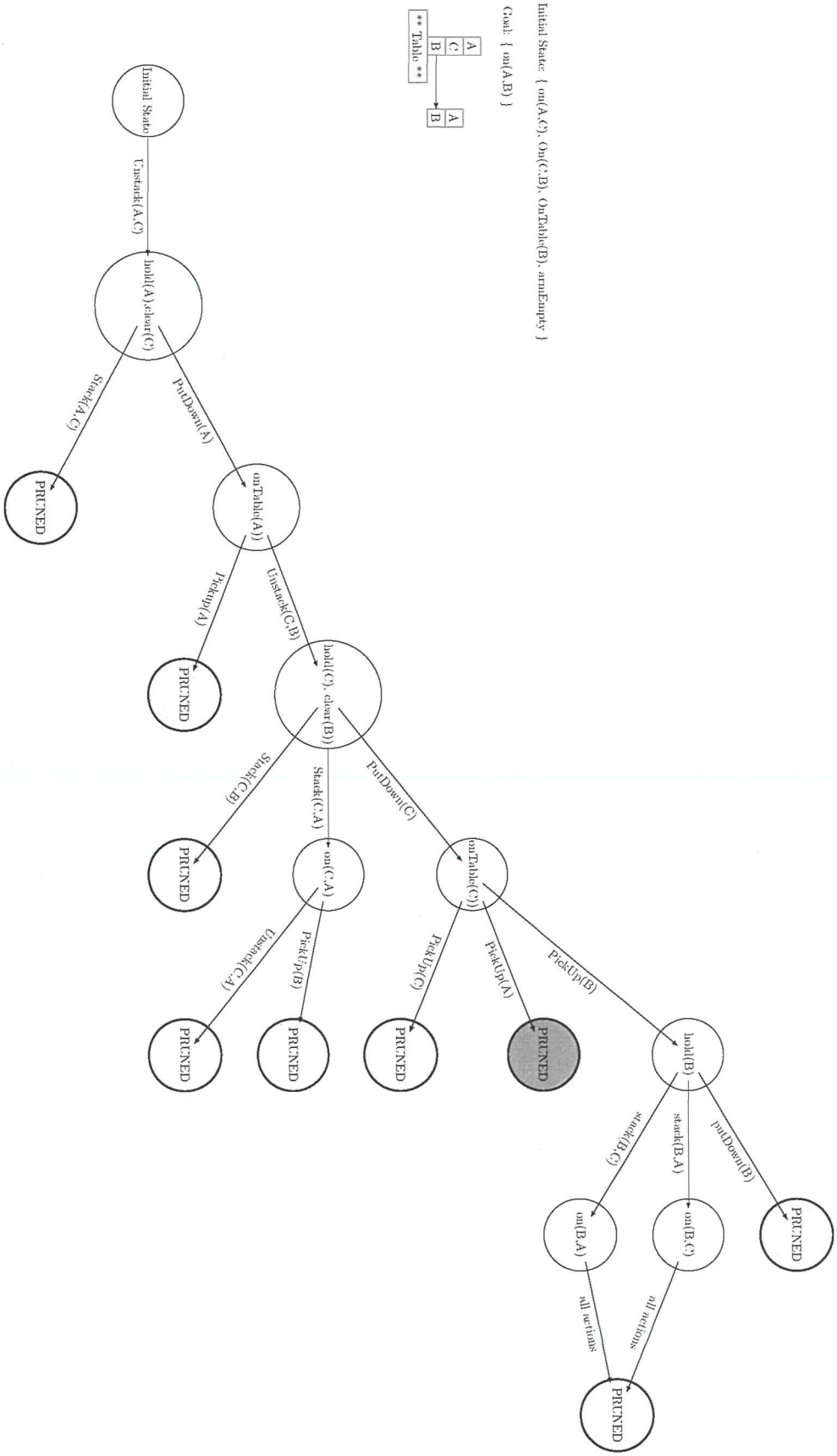


Figure 2: IW tree for question 2.b) Note that the pruned state with darker red is the node that was leading to the goal but was pruned because all its atomic fluents have been seen before. IW(2) would have continued that branch.

AI Planning for Autonomy

Sample Solutions for Problem Set VII: Value & Policy Iteration

1. We need to calculate the expected return for each action: pass or shoot.

   If Messi passes:

$$\begin{aligned} Q(Messi, Pass) &= P_{pass}(Suarez|Messi)[r(Messi, pass, Suarez) + \gamma \cdot V(Suarez)] \\ &= 1 \cdot [-1 + 1 \cdot -1.2] \\ &= 1 \cdot -2.2 \\ &= -2.2 \end{aligned}$$

   If Messi shoots:

$$\begin{aligned} Q(Messi, Shoot) &= P_{shoot}(Suarez|Messi)[r(Messi, shoot, Suarez) + \gamma \cdot V(Suarez)] + \\ &\quad P_{shoot}(Scored|Messi)[r(Messi, shoot, Scored) + \gamma \cdot V(Scored)] \\ &= 0.8[-2 + 1 \cdot -1.2] + 0.2[-2 + 1 \cdot 1.0] \\ &= -2.56 + (-0.2) \\ &= -2.76 \end{aligned}$$

   Therefore, to maximise our reward, Messi should pass.

2. To calculate $V(Messi)$, we choose the action that maximises our Q-value (expected future discounted reward):

$$\begin{aligned} V(Messi) &= \max(Q(Messi, pass), Q(Messi, shoot)) \\ &= \max(-2.2, -2.76) \text{ (from previous question)} \\ &= -2.2 \end{aligned}$$

   For *Scored*, there is only one action, which leads directly to the *Messi* state:

$$\begin{aligned} V(Scored) &= P_{return}(Messi|Scored)[r(Scored, return, Messi) + \gamma \cdot V(Messi)] \\ &= 1[2 + 1 \cdot -2.0] \\ &= 0 \end{aligned}$$

   For Suarez, the situation is similar to Messi:

$$\begin{aligned} V(Suarez) &= \max(Q(Suarez, pass), Q(Suarez, shoot)) \\ &= \max(P_{pass}(Messi|Suarez)[r(Suarez, pass, Messi) + \gamma \cdot V(Messi), \\ &\quad \Big(P_{shoot}(Messi|Suarez)[r(Suarez, shoot, Messi) + \gamma \cdot V(Messi) + \\ &\quad P_{shoot}(Scored|Suarez)[r(Suarez, shoot, Scored) + \gamma \cdot V(Scored)]\Big) \\ &= \max(1.0[-1 + 1 \cdot -2.0], (0.4[-2 + 1 \cdot 2.0] + 0.6[-2 + 1 \cdot 1.0])) \\ &= \max(-3, (0.4[-2 + 1 \cdot -2.0] + 0.6[-2 + 1 \cdot 1.0])) \\ &= \max(-3, (-1.6 + -0.6)) \\ &= -2.2 \end{aligned}$$

   Thus, the new table is:

Iteration		1	2	3	4
V(Messi)	=	0.0	-1.0	-2.0	-2.2
V(Suarez)	=	0.0	-1.0	-1.2	-2.2
V(Scored)	=	0.0	2.0	1.0	0.0

3. Policy Iteration has two main steps, policy evaluation and policy update. In order to evaluate the given policy:

$$
\begin{aligned}
V^{\pi}(Messi) &= Q^{\pi}(Messi, Pass) \\
&= P_{pass}(Suarez|Messi)[r(Messi, pass, Suarez) + \gamma \cdot V^{\pi}(Suarez)] \\
&= \gamma \cdot V^{\pi}(Suarez) - 1 \\
V^{\pi}(Suarez) &= Q^{\pi}(Suarez, Pass) \\
&= P_{pass}(Messi|Suarez)[r(Suarez, pass, Messi) + \gamma \cdot V^{\pi}(Messi)] \\
&= \gamma \cdot V^{\pi}(Messi) - 1 \\
V^{\pi}(Scored) &= Q^{\pi}(Scored, return) \\
&= P_{return}(Messi|Scored)[r(Scored, return, Messi) + \gamma \cdot V^{\pi}(Messi)] \\
&= \gamma \cdot V^{\pi}(Messi) + 2
\end{aligned}
$$

Then solve a very basic linear algebra about $V^{\pi}(Messi)$ and $V^{\pi}(Suarez)$:

$$
\begin{aligned}
V^{\pi}(Messi) &= 1/(\gamma - 1) \\
V^{\pi}(Suarez) &= 1/(\gamma - 1) \\
V^{\pi}(Scored) &= 3 + 1/(\gamma - 1)
\end{aligned}
$$

Then apply $\gamma = 0.8$, the policy evaluation table would be:

Iter	$Q^{\pi}(Messi, P)$	$Q^{\pi}(Messi, S)$	$Q^{\pi}(Suarez, P)$	$Q^{\pi}(Suarez, S)$	$Q^{\pi}(Scored)$
0	0	0	0	0	0
1	-5	-5.52	-5	-4.56	-2
2	-4.194	-4.772	-4.355	-3.993	-1.355

Then implement two iteration of policy update based on value from the policy evaluation table:

Iter	$\pi(Messi)$	$\pi(Suarez)$	$\pi(Scored)$
0	Pass	Pass	Return
1	Pass	Shoot	Return
2	Pass	Shoot	Return

$a = \gamma b - 1$

$b = \gamma a - 1$

$c = \gamma a + 2$

$b = \gamma(\gamma b - 1) - 1$

$b = \gamma^2 b - \gamma - 1$

$\gamma + 1 = \gamma^2 b - b = b(\gamma^2 - 1) = b(\gamma + 1)(\gamma - 1)$

$b = 1/(\gamma - 1)$

$c = \gamma(1/\gamma - 1) + 2$

$c = \frac{\gamma - 1 + 1}{\gamma - 1} + 2$

$c = 3 + \frac{1}{\gamma - 1}$

$V^{\pi}(M) = a = 1/\gamma - 1 = -5$

$V^{\pi}(S) = b = 1/\gamma - 1 = -5$

$V^{\pi}(Scored) = c = 3 + \frac{1}{\gamma - 1} = -2$

$Q^{\pi}(M, S) = P_{shoot}(Sw \mid Messi)[r(Messi, shoot, Suarez) + \gamma \cdot V^{\pi}(Suarez)]$

$+ P_{shoot}(Scored \mid Messi)[r(Messi, shoot, Scored) + \gamma V^{\pi}(Scored)]$

$= 0.8 \times (-2 + 0.8 \times -5) + 0.2 \times [-2 + 0.8 \times (-2)]$

$= 0.8 \times (-6) + 0.2 \times (-3.6)$

$= -4.8 + (-0.72) = -5.52 < -5 \Rightarrow$ pass

$Q^{\pi}(S, shoot) = P_{shoot}[0.6 \times [-2 + 0.8 \times (-2)] + 0.4 \times [-2 + 0.8 \times (-5)]$

$= -3.6 + (-2.4)$

-2.16

$= -4.56 > -5 \Rightarrow$ shoot

AI Planning for Autonomy

Sample Solutions for Problem Set VIII: Monte-Carlo Tree Search

1. MCTS tree updates[1], and the tree is included at the end of this document for each step:

   I1: Select $(2,1)$; Expand N; Do simulation on $\xrightarrow{succ}$ $(2,2)$; Backup the reward -1.

   I2: Select $(2,1) \rightarrow (2,2)$; Expand E; Do simulation on $\xrightarrow{slip(S)}$ $(1,2)$; Backup the reward -1.

   I3: Select $(2,1)$; Expand E; Do simulation on $\xrightarrow{succ}$ $(3,1)$; Backup the reward -1.

   I4: Select $(2,1)$; Expand W; Do simulation on $\xrightarrow{succ}$ $(2,1)$; Backup the reward -1.

   I5: Select $(2,1) \rightarrow (2,2)$; Expand S; Do simulation on $\xrightarrow{slip(W)}$ $(2,2)$; Backup the reward -1.

2. Calculate using $\text{argmax}_{a \in A} Q(s,a)$. The answer would be W (West) because it has the highest Q-value.

   W: $Q((2,1),W) = 0.8$

   E: $Q((2,1),E) = -0.8$

   N: $Q((2,1),N) = 0.08$

   S: $Q((2,1),S) = 0$

3. Need to calculate π for each of N, S, E, W based on the UCT formula and then normalise. However, in MCTS, normally we expand all the successors once before we run UCT forumla.

$$\pi(s) = argmax_{a \in A(s)} \begin{pmatrix} W & : & 0.8 + \sqrt{\frac{2 \ln 5}{1}} \\ E & : & -0.8 + \sqrt{\frac{2 \ln 5}{1}} \\ S & : & \infty \\ N & : & 0.08 + \sqrt{\frac{2 \ln 5}{3}} \end{pmatrix}$$

   Therefore, UCT would be more likely to choose S.

[1] The iteration traces in this workshop are generated by vanilla version MCTS, which are slightly different from the Lecture Slides (Select phase does not consider a not-fully expanded node as most urgent node to select in the vanilla version MCTS). However, if you choose any policy, such as UCB as Tree Policy (for selection and expansion), then based on that policy, the selection would select a not fully expanded node first.

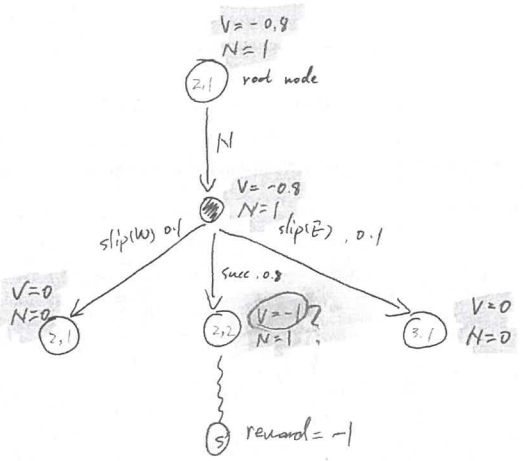


Figure 1: MCT after iteration 1

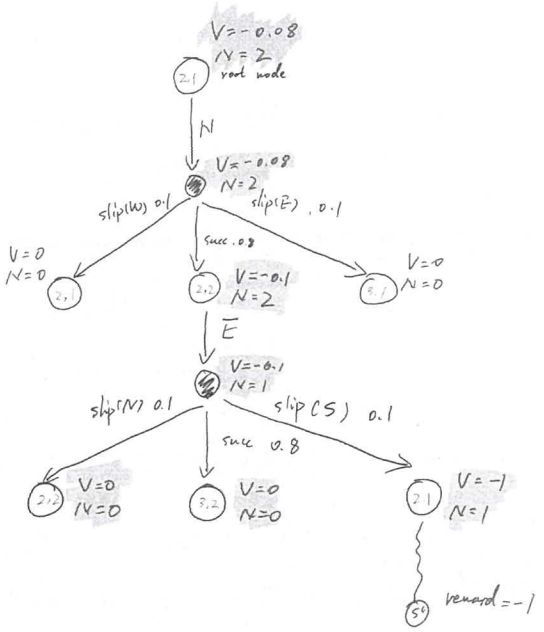


Figure 2: MCT after iteration 2

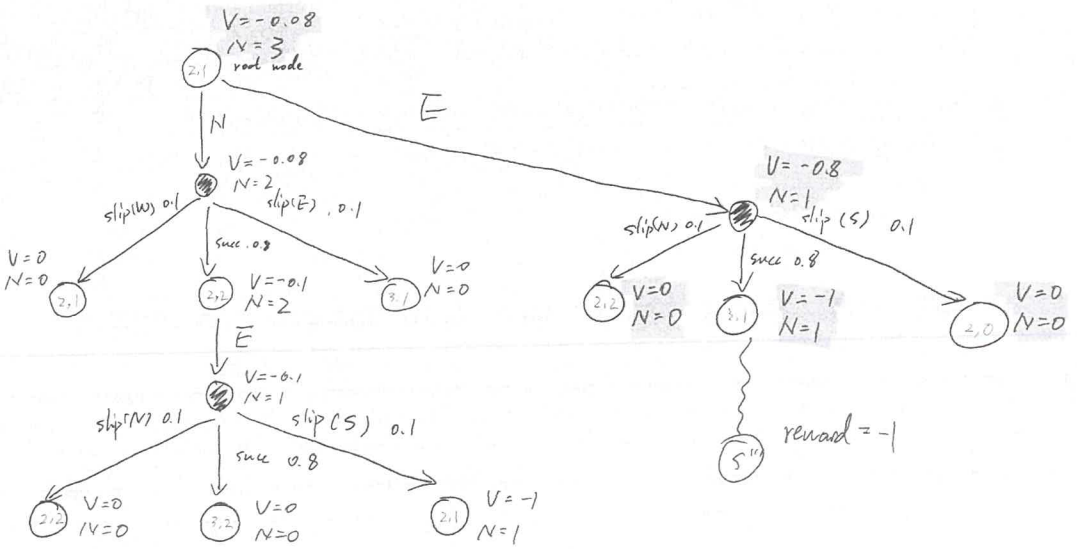


Figure 3: MCT after iteration 3

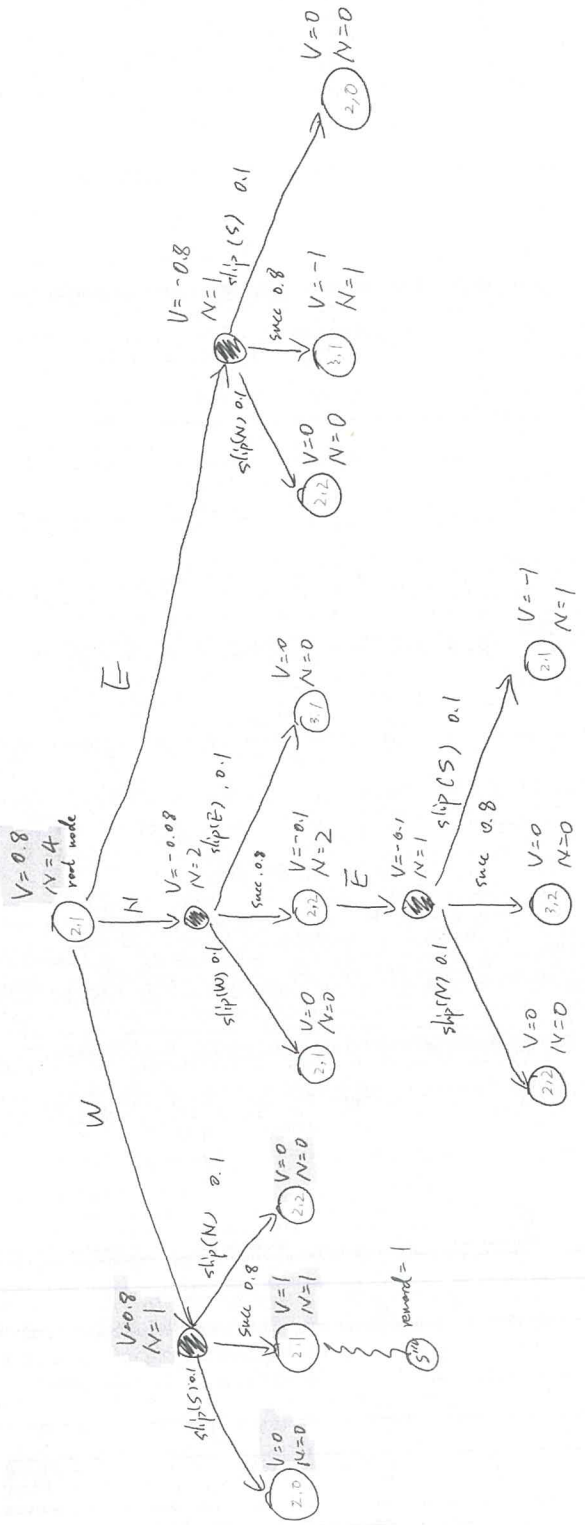


Figure 4: MCT after iteration 4

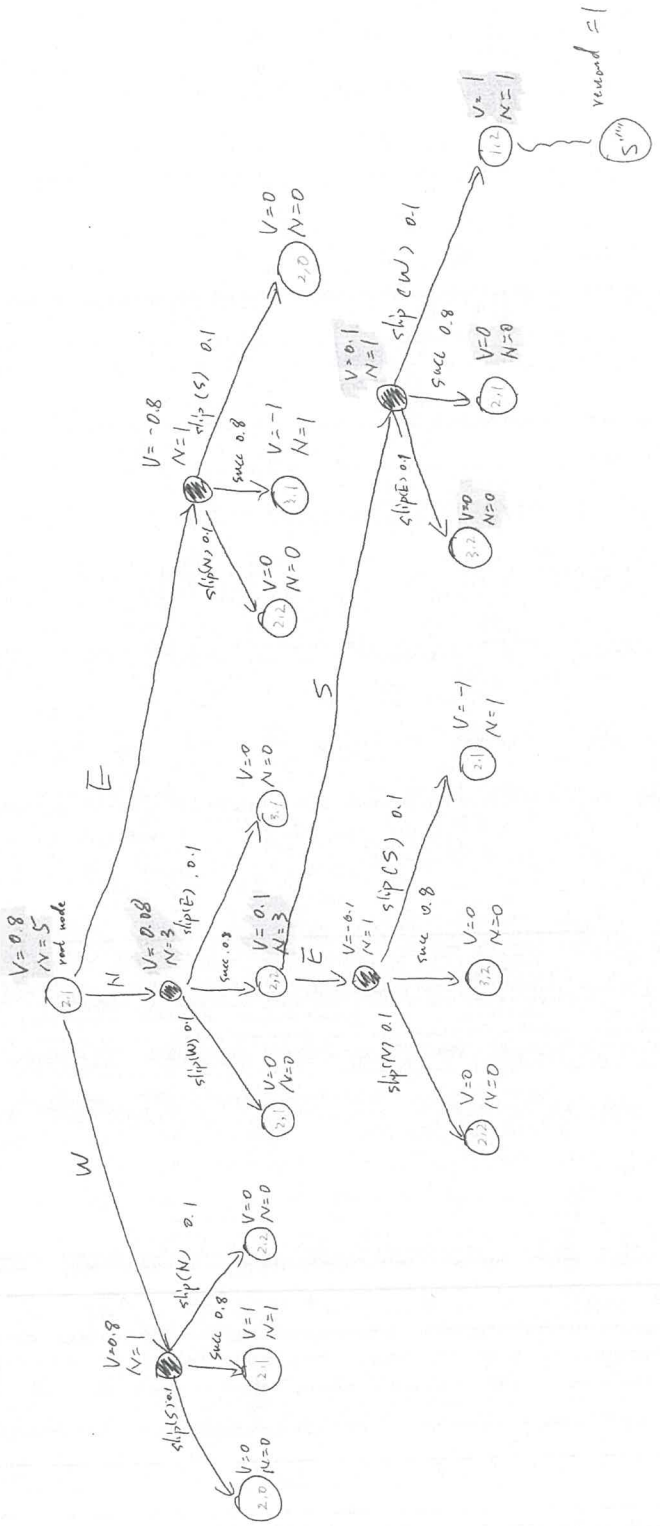


Figure 5: MCT after iteration 5

AI Planning for Autonomy

Sample Solutions for Problem Set IX: MDPs and Reinforcement Learning

1. The difference between SARSA and Q-learning is that Q-learning is "off-policy" learning, while Sarsa is "on-policy" learning. Essentially, this means that SARSA chooses its action using the same policy used to choose the previous action, and then uses this difference to update its Q-function, while Q-learning simply chooses the next value based on the maximum Q-value.

   Q-learning is therefore 'optimistic' in that when it updates, it assumes that in the next state, the 'best' (greedy) action will be chosen, even it may be that in the next step, the policy, such as ϵ-greedy, will choose to explore an action other than the best.

   SARSA instead knows the action that it will execute next when it performs the update, so will learn on the action whether it is best or not.

2. For Q-learning, this is calculated as:

$$\begin{aligned} Q(S,P) &= Q(S,P) + 0.4 \cdot [r(S,P) + 0.9 \cdot \max_{a' \in A(M)} Q(M,a') - Q(S,P)] \\ &= -0.7 + 0.4 \cdot [(-1) + 0.9 \cdot (-0.4) - (-0.7)] \\ &= -0.7 + 0.4 \times (-0.66) \\ &= -0.964 \end{aligned}$$

3. For SARSA, this is calculated as:

$$\begin{aligned} Q(S,P) &= Q(S,P) + 0.4 \cdot [r(S,P) + 0.9 \cdot Q(M,\pi(M)) - Q(S,P)] \\ &= -0.7 + 0.4 \cdot [(-1) + 0.9 \cdot (-0.8) - (-0.7)] \\ &= -0.7 + 0.4 \times (-1.102) \\ &= -1.108 \end{aligned}$$

4. For 3-step SARSA is calculated as:

$$Q(s,a) = Q(s,a) + \alpha[G_t^n - Q(s,a)]$$

$$G_t^n = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots + \gamma^n \cdot Q(S_{t+n}, \pi(S_{t+n}))$$

$$\begin{aligned} Q(S,P) &= Q(S,P) + 0.4 \cdot [G_S^3 - Q(S,P)] \\ &= -0.7 + 0.4 \cdot [-1.4716 - (-0.7)] \\ &= -0.7 + 0.4 \times (-0.7716) \\ &= -1.00864 \end{aligned}$$

   where

$$\begin{aligned} G_S^3 &= r(S,P) + 0.9 \cdot r(M,S) + 0.9^2 \cdot r(Scored,R) + 0.9^3 \cdot Q(M,P) \\ &= (-1) + 0.9 \cdot (-2) + 0.9^2 \cdot 2 + 0.9^3 \cdot (-0.4) \\ &= (-1) + (-1.8) + 1.62 + (-0.2916) \\ &= -1.4716 \end{aligned}$$

   And basically yes, it can converge much faster than 1-step.

AI Planning for Autonomy

Sample Solutions for Problem Set X: Game Theory

1. Having done Task 1 with your tutor, you will see that the outcome is that the winner is the person who values the item the most, they will (should!) be prepared to bid their true value, and in a good auction, they will pay somewhere between their value and the next highest value.

   This is the purpose of auctions, which are part of the greater field of *mechanism design*. Mechanism designers have used game theory for decades to design auctions and other mechanisms to elicit the truth from participants. That is, to define mechanisms in which the best response for all players is to bid their true value.

   Assume that you are participating in a classic English auction, in which people call out bids, with each bid higher than the last, until nobody bids any further. The winner is the final bidder.

   If you value the item at \$100, then your dominant strategy is to bid as low as possible, until you reach \$100, and then stop bidding. This is also the dominant strategy for all participants. If each participant is rational (and in real auctions, they are not!) and follows this strategy, then the winner will be the person who values the item the most, and they will pay just slightly more than what the second-highest bidder valued the item at. The same outcome as the Vickrey auction.

   Game theory has been used to show that many different types of single item auction, such as English auctions, Dutch auction, and Vickrey auctions, lead to the same outcomes.

2. The payoff matrix will look like this:

		Bidder B (val = 51)			
		49	50	51	52
Bidder A (val = 50)	49	$\frac{1}{2}$, 1	0, 2	0, 2	0, 2
	50	1, 0	0, $\frac{1}{2}$	0, 1	0, 1
	51	1, 0	0, 0	$-\frac{1}{2}$, 0	0, 0
	52	1, 0	0, 0	-1, 0	-1, $-\frac{1}{2}$

   For example, if Bidder A bids 50 and Bidder B bids 49, then Bidder B will receive 0 payoff (they won nothing), while Bidder A will receive 1, because they paid 49 for an item they value at 50. The difference is $50 - 49 = 1$, so this is their utility.

   a) Each bidder has a *weakly* dominant strategy. Consider Bidder A's reasoning process first:

      i. If Bidder B plays 49, then I should play 50, 51, or 52.

      If Bidder B plays 50, then I can play any strategy.

      If Bidder B plays 51, then I should play 49 or 50.

      If Bidder B plays 52, then I should play 49, 50, or 51.

      Thus, strategy 50 is the only (pure) strategy that is not dominated by some other (pure) strategy, so that is my weakly-dominant strategy.

ii. Bidder B then reasons in the same way to demonstrate that neither 51 is its weakly dominant strategy.

Note however, that Bidder B cannot start with the premise that Bidder A will choose 50 and then reason from here, because Bidder B does not know Bidder A's private value. This is an example of a game with *imperfect information*, which we will not consider any further in this subject.

b) Clearly, the combination of the two weakly dominant strategies is one equilibrium, but are their others? Yes!

The equilibria are at: (51, 49), (52, 49), (49, 50), (49, 51), (49, 52), (50, 51), (50, 52), (51, 50), (51, 52), (52, 50).

In each of these outcomes, neither player can improve their payoff by switching their strategy.

c) Assume that a player values the item at value v. Their bid b can be either $b = v$, $b < v$, or $b > v$.

If they decide to bid $b > v$ (overbid), then one of three outcomes can occur:

i. Another player bids higher than b. In this case, our player would have done just as well to bid v, because it still would have lost anyway and received utility of 0.

ii. All other players bid less than b. In this case, our player would have done just as well to bid v, because it would have won anyway and paid the next highest price.

iii. At least one other player bids b' such as $b > b' > v$. Now, our player wins, but pays b', which is more than it values the item, and thus its utility is $v - b' < 0$. So, it would have been better off bidding s and losing.

A similar argument can be made for under bidding. The parallel of the third case above it that there is some bidder who bids b' such that $v > b' > b$, and thus wins the auction. Our player could have won the auction by bidding v instead, and only paying b', thus gaining the utility $b' - v > 0$.

3. a) Having played the game with the tutor, what strategy did you choose?

Experiments have been done on this game, and, in the first game (payoff of 320) playing as the row play, most people play T essentially all the time. In the second game (payoff of 44) playing as the row player, most people play B essentially all the time.

b) Note that between the two games, the only payoff that changes is 320 to 44. Recall that in calculating mixed strategies, you should make your opponent indifferent to your moves, which means that the payoff for you is irrelevant to the probabilities that you should choose.

Therefore, to calculate the strategy that the row player *should play*, we can analyse both games at the same time, because our choice should be dependent purely on making the column player indifferent and the payoffs for the column player are the same in both games! Thus, as the row player, we should not change our strategy between the two games!

Consider the moves from the perspective of the column player:

$E(L) = 40X + 80(1 - X) = 80 - 40X$

$E(R) = 80X + 40(1 - X) = 40X + 40$

If $80 - 40X = 40 - 40X$ then $X = \frac{1}{2}$

Thus, in short, to make the column player indifferent, the row player should play both strategies with the same probability in both games.

If in game 1 you play top with $> 50\%$ probability, then you are falling right into column player's hands: the column player should simply play L with probability $\frac{1}{8}$ to ensure that you get the payoff of 40 most of the time.

$E(T) = 320Y + 40(1 - Y) = 280Y - 40$

$E(B) = 40Y + 80(1 - Y) = 80 - 40Y$

If $280Y - 40 = 80 - 40Y$ then $320Y = 40$ and $Y = \frac{1}{8}$ (and therefore $(1 - Y = \frac{7}{8})$.

I think this is awesome!