Problem 1: MDPs [15 pts.]

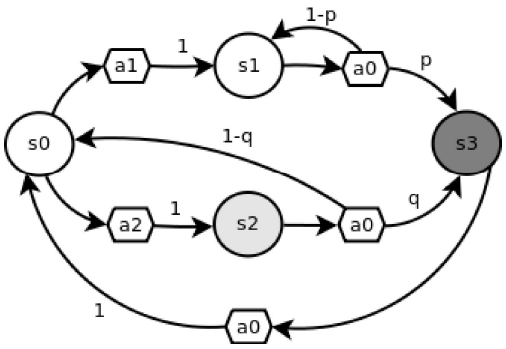

Figure 1: MDP for Problem 1. States are represented by circles and actions by hexagons. The numbers on the arrows from actions to states represent the transition probability, for instance, $P(s_3|s_2,a_0) = q$. Not shown are arrows for actions that have transition probability 0, for instance, $P(s_0|s_0,a_0) = 0$. Each of the parameters p and q are in the interval $[0,1]$. The reward is 10 for state s_3, 1 for state s_2 and 0 otherwise.

For this question, consider the infinite-horizon MDP $\mathcal{M}$ represented by Figure 1 with discount factor $\gamma \in [0,1)$. *Answers to this problem from Felipe Trevizan, TA for Grad AI in 2012.*

a) List all the possible policies for $\mathcal{M}$. [2 pts.]

Answer: There are two possible policies:

	s_0	s_1	s_2	s_3
π_1	a_1	a_0	a_0	a_0
π_2	a_2	a_0	a_0	a_0

b) Show the equation representing the optimal value function for each state of $\mathcal{M}$, i.e. $V^*(s_0), V^*(s_1), V^*(s_2)$ and $V^*(s_3)$. [2 pts.]

Answer:

$$
\begin{aligned}
V^*(s_0) &= \max_{a\in\{a_1,a_2\}} 0+\gamma\sum_{s'} P(s'|s,a)V^*(s') = \gamma\max\{V^*(s_1),V^*(s_2)\}\\
V^*(s_1) &= \max_{a\in\{a_0\}} 0+\gamma\sum_{s'} P(s'|s,a)V^*(s') = \gamma[(1-p)V^*(s_1)+pV^*(s_3)]\\
V^*(s_2) &= \max_{a\in\{a_0\}} 1+\gamma\sum_{s'} P(s'|s,a)V^*(s') = 1+\gamma[(1-q)V^*(s_0)+qV^*(s_3)]\\
V^*(s_3) &= \max_{a\in\{a_0\}} 10+\gamma\sum_{s'} P(s'|s,a)V^*(s') = 10+\gamma V^*(s_0)
\end{aligned}
$$

c) Is there a value for p such that for all $\gamma \in [0,1)$ and $q \in [0,1]$, $\pi^*(s_0) = a_2$? Explain. [4 pts.]

Answer: Notice that $\pi^*(s_0) = \text{argmax}_{a\in\{a_1,a_2\}} \gamma\sum_{s'} P(s'|s,a)V^*(s')$, therefore if $\gamma = 0$ then a_1 and a_2 are tied and π^* is not unique, so we can't guarantee that $\pi^*(s_0) = a_2$. For $\gamma > 0$, $\pi^*(s_0) = a_2$ iff $V^*(s_2) > V^*(s_1)$ for all $q \in [0,1]$. If $p = 0$, then $V^*(s_1) = \gamma V^*(s_1) = 0$ and since $V^*(s_2) = 1+\gamma[(1-q)V^*(s_0)+qV^*(s_3)] \geq 1$, we have that $V^*(s_2) > V^*(s_1)$ and $\pi^*(s_0) = a_2$ for all $\gamma \in (0,1)$ and $q \in [0,1]$.

d) Is there a value for q such that for all $\gamma \in [0,1)$ and $p > 0$, $\pi^*(s_0) = a_1$? Explain. [4 pts.]

Answer: No. Since $V^*(s_2) = 1+\gamma[(1-q)V^*(s_0)+qV^*(s_3)] \geq 1$, then $\pi^*(s_0) = a_1$ iff $V^*(s_1) > V^*(s_2) \geq 1$ for all $\gamma \in [0,1)$ and $p > 0$. We have that $V^*(s_1) = \gamma[(1-p)V^*(s_1)+pV^*(s_3)] = \frac{\gamma p V^*(s_3)}{1-\gamma(1-p)}$. Therefore we can always find a value for γ sufficiently small such that $V^*(s_1) < 1$. Notice that if γ equals 0, then π^* is not unique (as in the previous item).

e) Using $p = q = 0.25$ and $\gamma = 0.9$, compute π^* and V^* for all states of $\mathscr{M}$. You can either solve the recursion of item (b) or implement value iteration. In the latter case, the error between V^t and V^* allowed is $\epsilon = 10^{-3}$, therefore the stop criterion should be $\max_s |V^{t-1}(s) - V^t(s)| \leq \epsilon(1-\gamma)/\gamma$. [3 pts.]

Answer:

	s_0	s_1	s_2	s_3
V^*	14.1846	15.7608	15.6969	22.7661
π^*	a_1	a_0	a_0	a_0

Problem 2: Q-Learning [35 pts.]

You are to implement the Q-learning algorithm. Use a discount factor of 0.9. We have simulated an MDP-based grid world for you. The interface to the simulator is to provide a state and action and receive a new state and receive the reward from that state. The world is a grid of 10×10 cells, which you should represent in terms of the x and y coordinates of each cell. And there are four possible actions, North (N), South (S), East (E), and West (W). All the actions are non-deterministic. We have provided C++ versions of the simulator (available off the homework page, in both 32-bit and 64-bit form).

The C++ interface is `my_next_state(State, Action)` and `my_reward(State)`, where `State` and `Action` are defined in `mdp-simulation.h`. Instructions for using this file are also in the `.h` as comments.

You are to hand in the learned policy in terms of a function and a Q-table that we can query if needed. You are also to show a graph with the reward gathered by successive episodes of length 100 steps starting in some random position, as learning progresses.

I implemented the Q-learning algorithm on a 10 x 10 grid world with non-deterministic actions, 500 iterations per episode, and discount factor $\gamma = 0.9$. Since actions were non-deterministic, I used a different version of Q-learning than was presented in class. My update function, given a current state s, an action a that, when taken, led to new state s', is as follows

$$Q_n(s,a) \leftarrow (1-\alpha_n)Q_{n-1}(s,a) + \alpha_n \left[R(s') + \max_{a'} Q_{n-1}(s',a') \right]$$

with α_n defined as, for example, one of the following:

$$\alpha_n = \frac{1}{1 + visits_n(s,a)} \qquad \alpha_n = \frac{x}{y+n} \qquad \alpha_n = \frac{\log(n)}{n} \qquad \alpha_n = x \in [0,1]$$

This α_n represents the *learning rate*, which is essentially a weighting of how much the agent cares about its old beliefs versus whatever new belief is currently being presented. For more information, see the seminal paper by Watkins and Dayan [3].

The results, like all results in reinforcement learning, were affected by exploitation versus exploration in action selection. I used a mixture of three basic action selection functions: `maxAction`, `randomAction`, and `mixAction`. On the pure exploitation side, `maxAction` deterministically chooses the action with the highest current Q-value for our current state. On the pure exploration side, `randomAction` chooses any next action according to a uniform random distribution. Finally, as a basic mixture of the two, `mixAction` chooses a number $x \in [0,1)$, sees if $x < \epsilon$ for some $\epsilon \in [0,1)$, and executes either `randomAction` or `maxAction`. For my experiments, $\epsilon = 0.05$.

In Figure 2, we show the evolution of the average total discounted reward starting at each state in the grid as the number of episodes of Q-learning increases (rows), as well as across different action selection strategies (columns). In the first row, after 5 episodes, we see that neither the fully random action selection strategy (left column) nor the ϵ-greedy selection strategy (right column) provides a very accurate policy for start states that are far from the two main reward sinks. At 25 episodes, both strategies are starting to provide direction for states that are a medium distance from the two reward sinks. Finally, by 10,000 episodes, both strategies provide a decent approximation of the optimal policy. Critically, though, the purely random strategy is still missing direction for the farthest top right states.

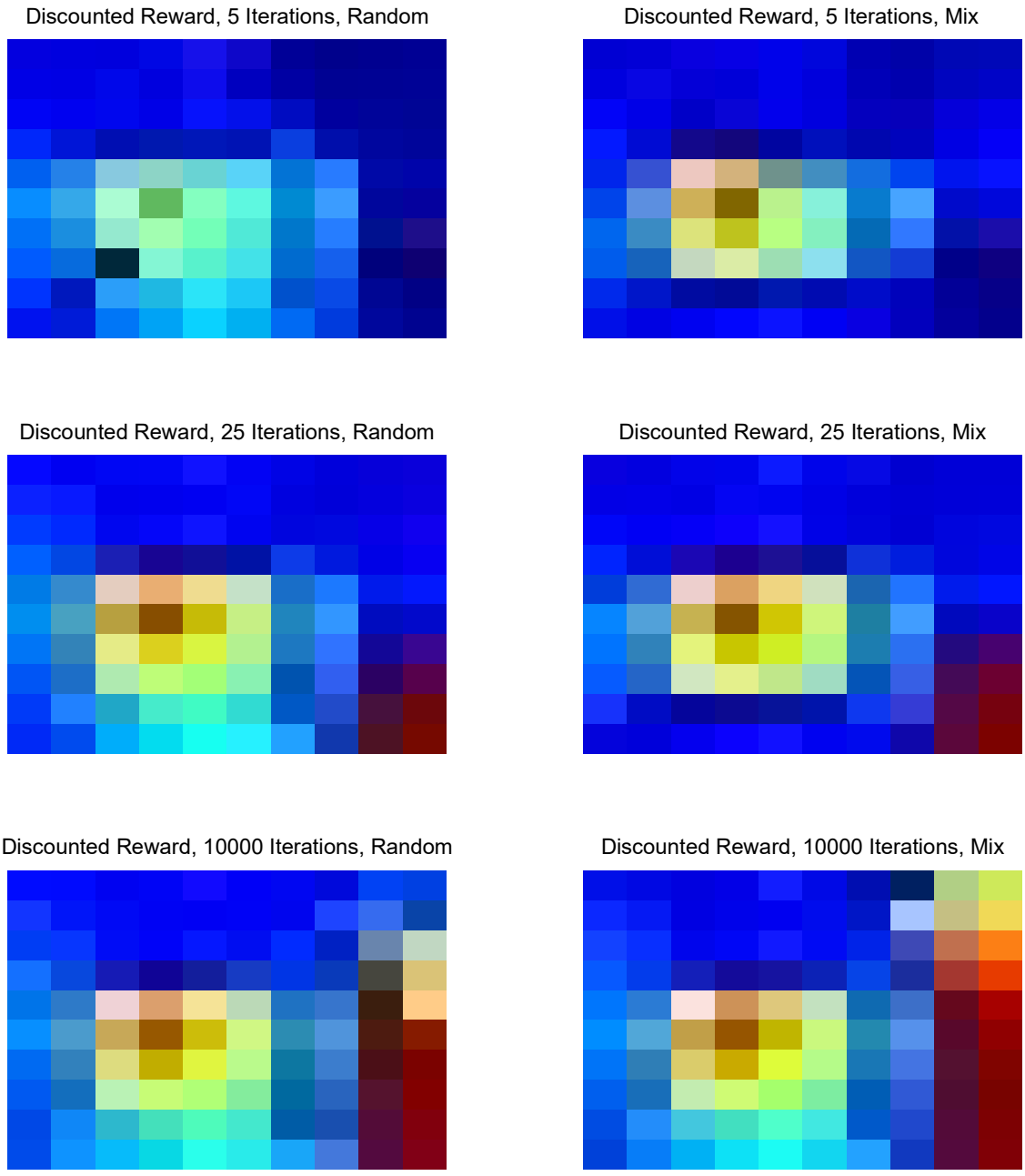

Figure 2: Comparison of Q-learning with two different action selection strategies. The left column represents selecting an action a with uniform probability $\frac{1}{|A|}$, while the right column represents the ϵ-greedy strategy of selecting the current max-Q action with probability ϵ (exploitation), and selecting an action randomly (exploration) with probability $1-\epsilon$. For these experiments, $\epsilon = 0.05$. Note that the ϵ-greedy strategy tends to converge more quickly.

In Figure 3, we show the average total discounted rewards obtained from executing 100 runs from each start state, using `maxAction` on our Q-table. There is some evident structure in the world—or my Q-learner is broken—on the rightmost states. On the right of Figure 3, we "cheat" a little and show the immediate rewards for each state

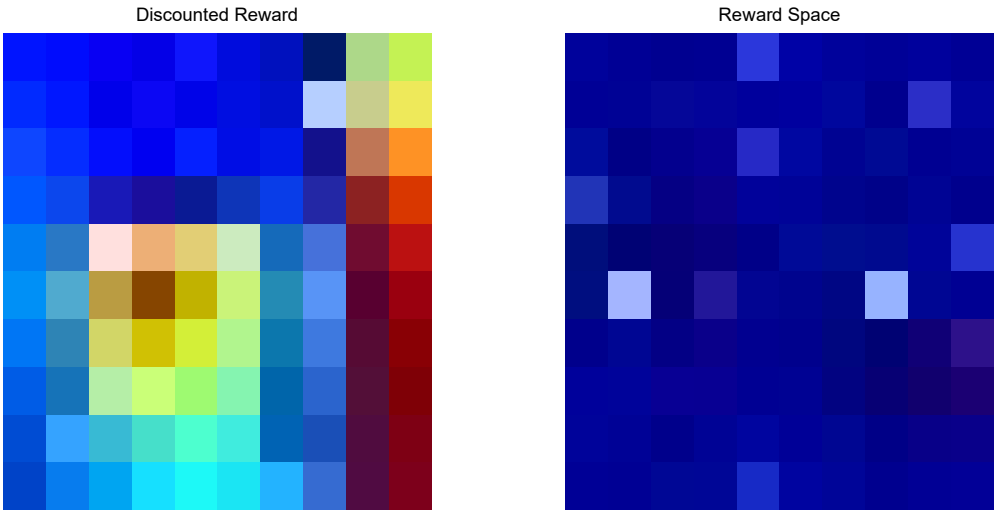

Figure 3: Left: Final heatmap showing average total discounted reward (100 runs on a policy formed from 100,000 Q-learning runs). Right: The true space of rewards in the grid world.

in the grid world.

Finally, Figure 4 presents the optimal policy found for this 10×10 grid world. This was computed after 100,000 episodes of Q-learning, each with 500 iterations starting from a random location in the space. We used ϵ-greedy action selection with $\epsilon = 0.05$, and α_n was defined as $\frac{1}{1+visits_n(s,a)}$. The other three α functions described above converged to 0 too quickly, which curtailed learning before the Q-table could converge to optimality.

E	E	N	N	W	S	S	S	E	E
E	E	S	W	W	S	E	N	E	E
E	E	S	N	W	S	E	E	E	E
E	S	S	S	W	N	E	S	E	E
N	E	N	E	E	S	N	E	E	E
N	N	N	W	S	S	N	N	E	S
W	W	W	W	S	S	W	W	E	W
W	W	W	W	S	S	W	W	E	W
W	N	N	N	W	S	W	W	E	W
W	N	N	N	W	S	S	S	N	W

Figure 4: Optimal policy for a 10×10 grid world with four non-deterministic actions.

Problem 3: POMDP Action Models [15 pts.]

Let the initial belief state b_0 for the 4x3 POMDP world be the uniform distribution over the nonterminal states, i.e., $\langle 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 0, 0\rangle$. We will calculate the exact belief state b_1 after the agent moves LEFT and its sensor reports exactly 1 adjacent wall using two possible realistic action models. For this question, you may either (a) show your calculations analytically or (b) write a small program/script to do it. *If you choose option (a), make sure to show all your work. If you choose option (b), please upload your code with the solution along with a read me file on how to run your code.*[1]

a) What is your sensor model, $P(e|s')$? [1 pt.]

b) Presume the action model states that every time that an action is against a wall, nothing happens and the agent/robot stays in the same state. Thus, for example, if we are in the state (1,1) and are looking only at the action LEFT, the action model is $P((1,1)|LEFT,(1,1)) = 1$. Under this model, calculate the exact belief state b_1 after the agent moves Left and its sensor reports exactly 1 adjacent wall. [6 pts.]

c) Presume the action model states that every time an action is against a wall, the probability of staying in the same state is additive over the possible cases. Thus, for example, if we are in the state (1,1) and take the action LEFT, the probability of hitting the wall to the left is 0.8 and the probability of hitting the wall to the right is 0.1. Thus, the total probability $P((1,1)|LEFT,(1,1)) = 0.9$. Now, under this model, calculate the exact belief state b_1 after the agent moves Left and its sensor reports exactly 1 adjacent wall. [6 pts.]

d) In each case, which location is the highest probability location? Are the two answers the same or different? [2 pts.]

Solutions: 3a

An acceptable sensor model is to use P(e = Observing EXACTLY 1 wall | s = certain (x,y) grid cell). If you use this model, then you get the sensor model matrix for the grid:

$$0,0,1,0$$
$$0,0,1,1$$
$$0,0,1,0$$

Another acceptable sensor model was the 0.9/0.1 sensor model described in the book where the sensor has 0.9 probability of reporting the correct answer (i.e. in (1,1), reporting there are 2 walls) and 0.1 probability of reporting the wrong number of walls (i.e. in (1,1) reporting that there is only 1 wall).

Other sensor models were accepted as well, and based on the assumptions students stated, problem scores were adjusted.

Solutions: 3b

This is the answer for the grid (using simulation) for the perfect sensor model:

$$0.0000, 0.0000, 0.0111, 0.0000$$
$$0.0000, 0.0000, 0.1333, 0.0111$$
$$0.0000, 0.0000, 0.1000, 0.0000$$

[1] This problem was adapted from AIMA 3ed., page 691.

Normalization was not required (since the answer doesn't change with normalization) but with normalization, the solution is:

$b(3,1) = \frac{9}{23}$
$b(3,2) = \frac{12}{23}$
$b(3,3) = \frac{1}{23}$
$b(4,2) = \frac{1}{23}$

Here is the answer of the 0.9/0.1 sensor model (with normalization):

$b(1,1) = \frac{18}{274}$
$b(2,1) = \frac{10}{274}$
$b(3,1) = \frac{81}{274}$
$b(4,1) = \frac{81}{274}$
$b(1,2) = \frac{10}{274}$
$b(3,2) = \frac{108}{274}$
$b(4,2) = \frac{9}{274}$
$b(1,3) = \frac{18}{274}$
$b(2,3) = \frac{10}{274}$
$b(3,3) = \frac{9}{274}$
$b(4,3) = 0$

Solutions: 3c

This is the answer for the grid (using simulation) without normalization:

$$0.0000, 0.0000, 0.0222, 0.0000$$
$$0.0000, 0.0000, 0.1111, 0.0111$$
$$0.0000, 0.0000, 0.1111, 0.0000$$

Once again, normalization was not required but the answer with normalization is:

$b(3,1) = \frac{10}{23}$
$b(3,2) = \frac{10}{23}$
$b(3,3) = \frac{2}{23}$
$b(4,2) = \frac{1}{23}$

Here is the answer of the 0.9/0.1 sensor model (with normalization):

$b(1,1) = \frac{9}{137}$
$b(2,1) = \frac{5}{137}$
$b(3,1) = \frac{45}{137}$
$b(4,1) = \frac{1}{274}$
$b(1,2) = \frac{5}{137}$
$b(3,2) = \frac{45}{137}$
$b(4,2) = \frac{9}{274}$
$b(1,3) = \frac{9}{137}$
$b(2,3) = \frac{5}{137}$

$b(3,3) = \frac{9}{137}$
$b(4,3) = 0$

Solutions: 3d

In 3(b), the highest probability location is (3,2). In 3(c), the highest probability location is a tie between (3,2) and (3,1). t is interesting that both 3(b) and 3(c) are plausible action transition models that could be potentially used on localizing a robotic system but the answer in localization is different.

Note: The same 3(d) is true for both the perfect and 0.9/0.1 sensor models.

Problem 4: POMDP Grid World Navigation [35 pts.]

In this problem, we will explore using POMDPs to plan for navigating in discrete grid worlds. While you will not have to actually solve the POMDPs, if you want to try to solve them, we suggest using the zMDP solver: `https://github.com/trey0/zmdp`.

(1) POMDP Localization

In this problem, the map of the world is known (see Figure 5), but the robot's position and orientation is unknown. The goal for the robot is to get to square `S31` and "announce" that it has reached the goal. If it announces and is actually at the goal location, it receives a reward of +100; if it announces and is not at the goal, the reward is -1000. After announcing, the robot is "magically" transported to a *sink state* (not illustrated) where all actions leave it in that state and it gets no further reward. The robot can also move forward, turn left, and turn right. Turning left or right is deterministic. Moving forward takes it to the square in front of it, unless that square is blocked by a wall, in which case the robot stays where it is. Walls are the outside boundary of the grid and the boundaries of the gray squares, and are represented as solid black lines. The robot can also observe what is in front of it at a cost of 10. The observation tells whether there is a wall in front with 90% accuracy.

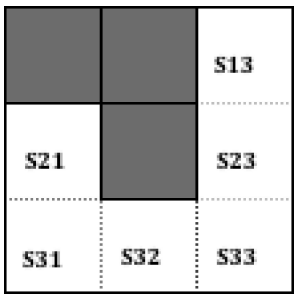

Figure 5: Our robot's world.

a) Write this domain as a POMDP. Include descriptions of the states, actions, transition function, observations, observation function, and rewards. Do not forget that all actions and observation functions must be defined for all states. Use the 3×3 grid above as the environment that the robot needs to explore. For cases where the transition or observation functions are repetitive, you can just enumerate the functions for one set of states and point out what the pattern is for the other states. [5 pts.]

b) Assume that the robot's initial belief is uniformly distributed amongst all the white cells in the environment, but it knows it is facing north (up, in the diagram). Assume that the robot executes the sequence ⟨forward;

right; observe; left; forward; observe⟩. Assume the first observation returns "wall" and the second returns "open". Show the belief states of the robot after each action. [5 pts.]

c) Show the optimal MDP policy (it is the obvious thing to do—you can easily generate it by hand). Using this policy and the assumption about the initial state given in (2) above, answer the following (in case of ties, choose an arbitrary, but fixed, tie-breaking method, and indicate which method you are using, such as choosing actions that come earlier in the dictionary): [10 pts.]

(a) What is the sequence of actions chosen (up to the "announce") using the "most likely" heuristic?

(b) What is the sequence of actions chosen (up to the "announce") using the "voting" heuristic?

(c) If the sequences differ, explain, in general terms, why the two heuristics produce different results; if the sequences are the same, explain, in general terms, what it is about this particular domain that they produce the same results.

(2) POMDP Mapping

In this problem, the robot knows its initial starting position and orientation, but it does not know which environment it is in (out of the six environments shown in Figure 6). The goal is for the robot to "announce" which environment it is in. If it announces correctly, it gets a reward of +100; if it announces incorrectly, the reward is -1000. After announcing, the robot is "magically" transported to a *sink state* (not illustrated) where all actions leave it in that state and it gets no further reward. The robot can also move forward, turn left, and turn right. Turning left or right is deterministic. Moving forward takes it to the square in front of it, unless that square is blocked by a wall, in which case the robot stays where it is. The robot can also observe what is in front of it at a cost of 10. The observation tells whether there is a wall in front with 100% accuracy. As in part 1, walls (solid black lines) appear at the borders of the grid and surrounding all gray squares.

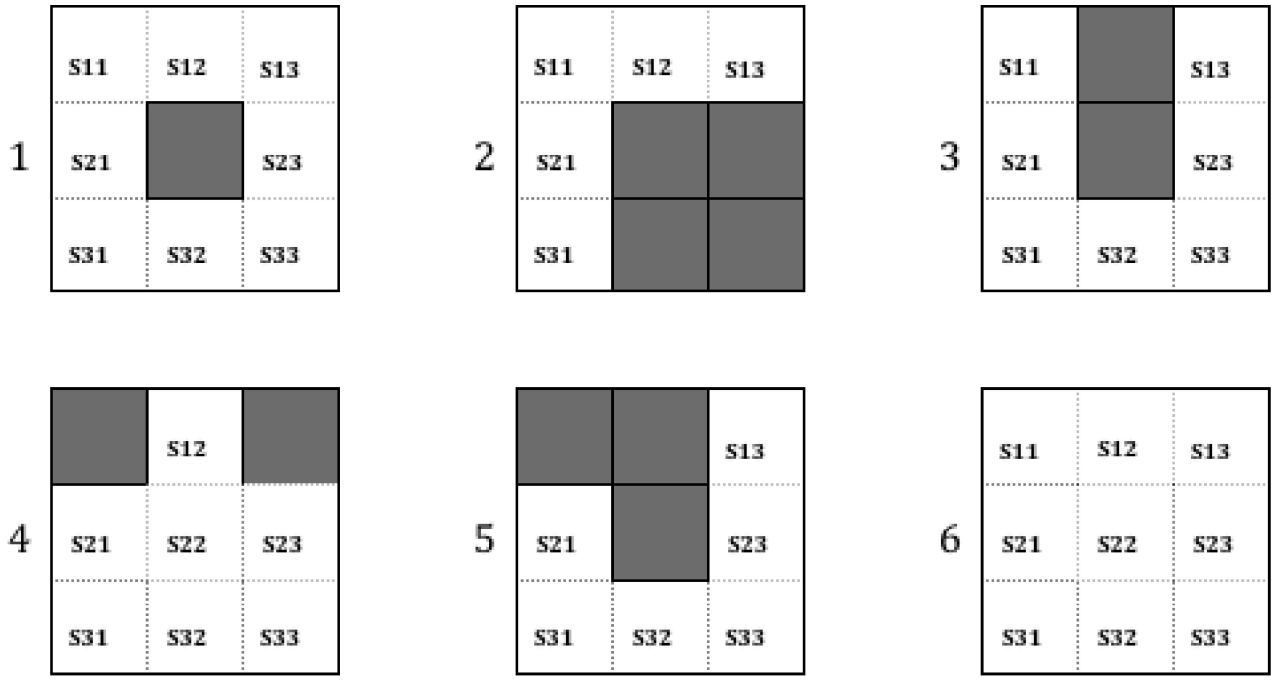

Figure 6: Uncertainty over six possible worlds in which our robot might reside.

a) Write this domain as a POMDP. Include descriptions of the states, actions, transition function, observations, observation function, and rewards. Use the six environments above as the set of environments that the

robot could possibly be in. For cases where the transition or observation functions are repetitive, you can just enumerate the functions for one set of states and point out what the pattern is for the other states. [5 pts.]

b) Assume that the robot's initial belief is that it is in state S31 facing north (up), and it has uniform belief about which environment it is in. Assume that the robot executes the sequence ⟨forward; right; observe; forward; left; forward; right; observe; announce environment 1 (the top left map above)⟩. Assume the first observation returns "wall" and the second returns "open". Show the belief states of the robot after each action. What is the expected, undiscounted reward of this sequence? [10 pts.]

Solutions Follow!

A. POMDP Localization

For this problem, we assume the map of the world is known, but the location and orientation of the robot are not.

a) A POMDP is defined as some tuple (S, A, O, T, Ω, R) with:

Symbol	Name	In Our Example
S	set of states	{grid × agent's orientation} ∪ {sink}
A	set of actions	{left, right, forward, announce, observe}
O	set of observations	{wall-in-front}
$T : S \times A \times S \to [0,1]$	transition function	probability of (not) moving agent to states
$\Omega : O \times S \times A \times S \to [0,1]$	observation function	given (¬)wall-in-front, where am I?
$R : S \times A \to \mathbb{R}$	reward function	announce (+100 or -1000), observe (-10)

We define the states to be $\{\{S21, S31, S32, S23, S33, S13, \text{sink}\} \times \{N, S, E, W\}\}$; that is, the state space is the set of all physical grid spaces cross all possible orientations in the space. We assume that the set of possible actions and observations are the same for all states. Then, we must define transition function T, observation function Ω, and reward function R for each state.

Sink states: The robot is transported to this state after announcing. The robot stays in this state regardless of action and observation and receives no reward. Thus

$$T(\text{sink}, \cdot, s) = \begin{cases} 1 & \leftarrow \quad s = \text{sink} \\ 0 & \leftarrow \quad s \neq \text{sink} \end{cases}$$

$$R(\text{sink}, \cdot) = 0$$

$$\Omega(o, \text{sink}, a, s) = \begin{cases} 0.1 & \leftarrow \quad s = \text{sink} \wedge a = \text{observe} \wedge o = \text{wall} \\ 0.9 & \leftarrow \quad s = \text{sink} \wedge a = \text{observe} \wedge o \neq \text{wall} \\ 0.5 & \leftarrow \quad s = \text{sink} \wedge a \neq \text{observe} \wedge o = \cdot \\ \textit{undefined} & \leftarrow \quad s \neq \text{sink} \end{cases}$$

Where sink represents the robot, with any orientation, existing in the sink state. Note Ω specifically: if we observe in sink and transition to sink, we have a 10% chance of believing a wall is nearby (although no wall exists) and a 90% chance of observing the true "no wall" state. If we do any other action in sink and transition to sink, we get no information, or 50%, about our one observation. Finally, if we do any action at all in sink and transition to some $s \neq$ sink, the behavior is undefined; this should never occur.

Goal states: State S31 is a goal state. The robot gets +100 reward if it performs announce here. State S31 has walls to the south and the west.

Transition Function			
State	Action	State'	Probability
S31-N,S,E,W	announce	sink	1.0
S31-N,S,E,W	observe	S31-N,S,E,W	1.0
S31-N	left	S31-W	1.0
S31-W	left	S31-S	1.0
S31-S	left	S31-E	1.0
S31-E	left	S31-N	1.0
S31-N	right	S31-E	1.0
S31-E	right	S31-S	1.0
S31-S	right	S31-W	1.0
S31-W	right	S31-N	1.0
S31-N	forward	S21-N	1.0
S31-E	forward	S32-E	1.0
S31-S	forward	S31-S	1.0
S31-W	forward	S31-W	1.0

The only interesting part of S31's observation function is that, upon execution of announce, the agent is nondeterministically transported to the sink state.

Reward Function		
State	Action	Reward
S31-N,S,E,W	announce	+100
S31-N,S,E,W	observe	-10
S31-N,S,E,W	{left, right, forward}	0

As mentioned before, the agent receives +100 reward if it announces in the goal state. It receives the standard -10 reward if it observes, and no reward otherwise.

Observation Function				
Observation	State	Action	State'	Probability
wall	S31-N,S,E,W	¬observe	·	0.5
¬wall	S31-N,S,E,W	¬observe	·	0.5
wall	S31-W,S	observe	·	0.9
¬wall	S31-W,S	observe	·	0.1
wall	S31-N,E	observe	·	0.1
¬wall	S31-N,E	observe	·	0.9

We receive no information (wall = 0.5, ¬wall = 0.5) about the only observation if we do not execute observe. If we are facing a wall in S31-W or S31-S, we receive a wall confirmation with probability 0.9. If we are not facing a wall in S31-N or S31-E, we receive a ¬wall confirmation with probability 0.9.

Non-goal states: States S21, S32, S23, S33, and S13 are non-goal, non-sink states. These will all have similar behavior to the goal state S31 above, except any announce action will result in -1000 reward before transporting the agent to the sink state. Specifically, the transition function is identical—with replacement of states—to S31. The reward function is as follows:

Reward Function		
State	Action	Reward
S21,32,23,33,13-N,S,E,W	announce	-1000
S21,32,23,33,13-N,S,E,W	observe	-10
S21,32,23,33,13-N,S,E,W	{left, right, forward}	0

Finally, the observation function for each state will follow the same spirit as the S31 observation function above. Below is S13's observation function as an example:

Observation Function				
Observation	State	Action	State'	Probability
wall	S13-N,S,E,W	¬observe	·	0.5
¬wall	S13-N,S,E,W	¬observe	·	0.5
wall	S13-N,E,W	observe	·	0.9
¬wall	S13-N,E,W	observe	·	0.1
wall	S13-S	observe	·	0.1
¬wall	S13-S	observe	·	0.9

The other states follow, adjusting for the presence or lack of walls.

b) Assume the robot assigns 1/6 probability to being in each state, but knows it is facing N. It executes action sequence {forward, right, observe, left, forward, observe}. The first observe returns wall; the second observe returns ¬wall. In the discussion of belief states below, assume 0% belief for all unmentioned states.

initial

S21-N	S31-N	S32-N	S23-N	S33-N	S13-N
1/6	1/6	1/6	1/6	1/6	1/6

forward

S21-N	S32-N	S33-N	S13-N
2/6	1/6	1/6	2/6

right

S21-E	S32-E	S33-E	S13-E
2/6	1/6	1/6	2/6

observe returns wall. Note that S21-E, S-33E, S13-E all have a wall, while S32-E does not. S32-E keeps 0.1 of its mass, while the three other states split 0.9 of S32-E's mass.

S21-E	S32-E	S33-E	S13-E
23/60	1/60	13/60	23/60

left

S21-N	S32-N	S33-N	S13-N
23/60	1/60	13/60	23/60

forward

S21-N	S32-N	S13-N
23/60	1/60	36/60

observe returns ¬wall. Note that each of S21-N, S32-N, S13-N has a wall. None of the non-zero states do not have a wall, so although ¬wall was returned, nothing changes.

S21-N	S32-N	S13-N
23/60	1/60	36/60

c) One way to avoid the computational intractability of solving POMDPs exactly is to use greedy heuristics to solve the underlying MDPs approximately. First, we give the optimal policy for the underlying MDP of the problem above. To break ties, we favor moving forward before turning clockwise before announcing.

Optimal MDP Policy							
State	Action	State	Action	State	Action	State	Action
S21-N	right	S21-E	right	S21-S	forward	S21-W	left
S31-N	announce	S31-E	announce	S31-S	announce	S31-W	announce
S32-N	left	S32-E	right	S32-S	right	S32-W	forward
S23-N	left	S23-E	right	S23-S	right	S23-W	forward
S33-N	right	S33-E	right	S33-S	forward	S33-W	left
S13-N	right	S13-E	right	S13-S	forward	S13-W	left
sink-N	·	sink-E	·	sink-S	·	sink-W	·

i) Using the "most likely" heuristic [1], we select the action with best Q-value for the most probable state.

- We start with 1/6 belief for facing north in each of the six Sxy-N states, and 0 belief in sink. We now run into a problem. Our policy choice is defined as

$$\pi(b(s)) = \underset{a}{\text{argmax}}\left(Q(\underset{s}{\text{argmax}}(b(s), a))\right)$$

  The issue is that $b(s) = 1/6$ for six states. We break this tie by choosing a state that is earlier in the dictionary, in this case S13-N. The policy then gives us action right.
- We now have 1/6 belief for facing east in the six Sxy-E states. Using the same tie-breaker, we choose action right again.
- Again, we have 1/6 belief for facing south in the six Sxy-S states. Tie-breaking to S13-S, we choose forward.
- Our belief is now 2/6 for S31-S, 1/6 for S32-S, 1/6 for S33-S, and 2/6 for S23-S. We tie break between S31-S and S23-S by choosing the lexicographically lower S23-S. The policy gives us right.
- Again, our belief is now 2/6 for S31-W, 1/6 for S32-W, 1/6 for S33-W, and 2/6 for S23-W. We tie break between S31-W and S23-W by choosing the lexicographically lower S23-W. The policy gives us forward.
- Our belief is now 3/6 S31-W, 1/6 S33-W, and 2/6 S32-W. We have no tie to break, so we choose to end with announce.

Thus, the final **most likely** sequence is: {right, right, forward, right, forward, announce}.

ii) Using the "voting" heuristic [2], we take the action that is represented by the most states, weighted by the belief that our agent is in that state.

- We start with 1/6 belief for facing north in each of the six main states, and 0 belief for the sink. Our policy has 3 right, 2 left, and 1 announce, so we choose right (with weight $3/6 > 2/6 > 1/6$).
- Now we have 1/6 belief for facing east in each of the six states. Our policy has 5 right, 1 announce, so we choose right (with weight $5/6 > 1/6$).
- Now we have 1/6 belief for facing south in each of the six states. Our policy shows 3 forward, 2 right, and 1 announce. We vote forward.
- We have 2/6 belief in S31-S, 1/6 in S32-S, 1/6 in S33-S, and 2/6 in S23-S. This yields 2/6 score for announce, $1/6 + 2/6 = 3/6$ for right, and 1/6 for forward. We vote for right.
- We have 2/6 belief in S31-W, 1/6 in S32-W, 1/6 in S33-W, and 2/6 in S23-W. This yields 2/6 score for announce, $1/6 + 2/6 = 3/6$ for forward, and 1/6 for left. We vote for forward.
- We have 3/6 belief in S31-W, 2/6 in S32-W, and 1/6 in S33-W. This yields 3/6 score for announce, 2/6 for forward, and 1/6 for left. We vote on announce and end.

The final sequence of actions for **voting** is: {right, right, forward, right, forward, announce}.

iii) In general, these two heuristics will not produce the same sequence of actions for a domain. In this case, they did; however, had I chosen a different tie breaking rule for equal beliefs in different states,

the results would have been quite different. For instance, had the tie breaker been "I will always choose S31 over other equal-belief states," the action sequence returned by the "most likely" heuristic would have been {announce}, while the "voting" heuristic would have remained unchanged.

It seems like the "voting" heuristic is more robust than the "most likely" heuristic, at only a slightly higher computational cost.

B. POMDP Mapping

For this problem, the robot knows his initial position and orientation, but does not know in which of six worlds $\{W1, \ldots, W6\}$ it is located.

a) Our new POMDP is defined as a tuple (S, A, O, T, Ω, R). The general O, T, Ω, and R are the same as in the previous problem, while the state space S and action space A are edited slightly to include the introduction of six different worlds:

$$\begin{aligned} S &= \{\{W1, W2, W3, W4, W5, W6\} \\ &\quad \times \{S11, S12, S13, S21, S22, S23, S31, S32, S33\} \\ &\quad \times \{N, S, E, W\}\} \\ &\quad \cup \{\{\text{sink}\} \times \{N, S, E, W\}\} \\ A &= \{\text{left}, \text{right}, \text{forward}, \text{observe}, \text{announce1}, \ldots, \text{announce6}\} \end{aligned}$$

Symbol	Name	In Our Example
S	set of states	defined above
A	set of actions	defined above
O	set of observations	{wall-in-front}
$T : S \times A \times S \to [0, 1]$	transition function	probability of (not) moving agent to states
$\Omega : O \times S \times A \times S \to [0, 1]$	observation function	given (¬)wall-in-front, where am I?
$R : S \times A \to \mathbb{R}$	reward function	announce (+100 or -1000), observe (-10)

Sink states: Defined as in Part A. Actions always lead to no new states, no new observations, no new positive or negative reward. Regardless of from which environment the robot announced, it is transported to this ubiquitous sink state.

Rewards: The goal states for a robot depend on in which environment the robot is located. For instance, let us assume that the robot is in environment W1. Then, for any of the legal states $s \in \{W1 - S11, W1 - S12, \ldots, W1 - S33\}$, if the robot announces W1, it will receive +100 reward before being transported to a sink state. Otherwise, for any of the legal states $s \in$ {W(2,3,4,5,6)-S11, …, W(2,3,4,5,6)-S33}, if the robot announces W1, it receives -1000 reward before sink state transportation.

Reward Function		
State	Action	Reward
W1-S11,...,S33-N,S,E,W	announce1	+100
W2-S11,...,S33-N,S,E,W	announce1	-1000
W3-S11,...,S33-N,S,E,W	announce1	-1000
W4-S11,...,S33-N,S,E,W	announce1	-1000
W5-S11,...,S33-N,S,E,W	announce1	-1000
W6-S11,...,S33-N,S,E,W	announce1	-1000
W1-S11,...,S33-N,S,E,W	announce2	-1000
W2-S11,...,S33-N,S,E,W	announce2	+100
W3-S11,...,S33-N,S,E,W	announce2	-1000
⋮	⋮	⋮
(W1,2,3,4,5,6)-S11,...,S33-N,S,E,W	observe	-10
(W1,2,3,4,5,6)-S11,...,S33-N,S,E,W	{left, right, forward}	0

Transitions: The transition functions for each state are similar to those defined in Part A. Critically, no action can ever warp the robot from one environment W_n to some other environment W_m—with the exception of announce, which nondeterministically moves the robot to the sink state.

Below is an example transition function, for state S21 in world W1. Note that the agent moves as in Part A, and never changes from world W1 to some other $w \in \{W2, W3, W4, W5, W6\}$.

Transition Function			
State	Action	State'	Probability
W1-S21-N,S,E,W	announce1...6	sink	1.0
W1-S21-N,S,E,W	observe	W1-S21-N,S,E,W	1.0
W1-S21-N	left	W1-S21-W	1.0
W1-S21-W	left	W1-S21-S	1.0
W1-S21-S	left	W1-S21-E	1.0
W1-S21-E	left	W1-S21-N	1.0
W1-S21-N	right	W1-S21-E	1.0
W1-S21-E	right	W1-S21-S	1.0
W1-S21-S	right	W1-S21-W	1.0
W1-S21-W	right	W1-S21-N	1.0
W1-S21-N	forward	W1-S11-N	1.0
W1-S21-E	forward	W1-S21-E	1.0
W1-S21-S	forward	W1-S31-S	1.0
W1-S21-W	forward	W1-S21-W	1.0

Observations: The robot's ability to observe has improved since Part A. It now deterministically returns whether or not a wall exists in front of the robot; however, its cost of -10 reward remains the same. This observation function is shared across all worlds and states except the mysterious sink state, where executing observe costs nothing.

Below is an example observation function, one for S21 in W1 and one for S21 in W5.

Observation Function				
Observation	State	Action	State'	Probability
wall	W1-S21-N,S,E,W	¬observe	·	0.5
¬wall	W1-S21-N,S,E,W	¬observe	·	0.5
wall	W5-S21-N,S,E,W	¬observe	·	0.5
¬wall	W5-S21-N,S,E,W	¬observe	·	0.5
wall	W1-S21-E,W	observe	·	1.0
¬wall	W1-S21-E,W	observe	·	0.0
wall	W1-S13-N,S	observe	·	0.0
¬wall	W1-S13-N,S	observe	·	1.0
wall	W5-S21-N,E,W	observe	·	1.0
¬wall	W5-S21-N,E,W	observe	·	0.0
wall	W5-S13-S	observe	·	0.0
¬wall	W5-S13-S	observe	·	1.0

Again, we see that performing an action that is not observe gives no information (0.5 wall, 0.5 ¬wall) about our only observation, while performing observe yields perfect information.

b) Finally, we compute a sample execution in this environment. The robot knows its initial state and orientation to be S31 and N; however, it does not know its world. It holds uniform belief of 1/6 that it is in each of {W1-S31-N, …, W6-S31-N}. It then executes {forward, right, observe, forward, left, forward, right, observe, announce1}. Assume the first observe returns wall and the second observe returns ¬wall.

(a) We show the belief state of the robot as it executes this sequence.

initial

W1-S31-N	W2-S31-N	W3-S31-N	W4-S31-N	W5-S31-N	W6-S31-N
1/6	1/6	1/6	1/6	1/6	1/6

forward

W1-S21-N	W2-S21-N	W3-S21-N	W4-S21-N	W5-S21-N	W6-S21-N
1/6	1/6	1/6	1/6	1/6	1/6

right

W1-S21-E	W2-S21-E	W3-S21-E	W4-S21-E	W5-S21-E	W6-S21-E
1/6	1/6	1/6	1/6	1/6	1/6

observe returns wall. Unlike Part A, this observation tells us with 100% certainty that there is a wall in front of our robot, regardless of his world. This completely rules out W4 and W6, so we distribute 2/6 probability mass evenly to the other four worlds.

W1-S21-E	W2-S21-E	W3-S21-E	W5-S21-E
1/4	1/4	1/4	1/4

forward. This is an odd forward, since we know a wall exists in front of us.

W1-S21-E	W2-S21-E	W3-S21-E	W5-S21-E
1/4	1/4	1/4	1/4

left

W1-S21-N	W2-S21-N	W3-S21-N	W5-S21-N
1/4	1/4	1/4	1/4

forward. W5 is blocked by a wall, all others move N one space.

W1-S11-N	W2-S11-N	W3-S11-N	W5-S21-N
1/4	1/4	1/4	1/4

right

W1-S11-E	W2-S11-E	W3-S11-E	W5-S21-E
1/4	1/4	1/4	1/4

observe returns ¬wall. W3-S11-E and W5-S21-E with observe would return wall, so they are ignored. Their probability mass of 1/2 is redistributed to the two remaining worlds W1 and W2.

W1-S11-E	W2-S11-E
1/2	1/2

announce1. Regardless of where the robot actually is, it is going straight to the sink.

sink-N,S,E,W
1.0

(b) If the above is correct, the robot has a 50% chance of announcing its environment correctly, and a 50% chance of being incorrect. Without any discounting, we expect the reward to be

$$\begin{aligned} E[Reward(b(S), \text{announce1}] &= \sum_{s \in S} b(s) \cdot R(s, \text{announce1}) \\ &= 0.5 \cdot (100 - 20) + 0.5 \cdot (-1000 - 20) \\ &= -470 \end{aligned}$$

We safely ignore any 0% states in the summation above.

Bibliography

[1] I. Nourbakhsh, R. Powers, and S. Birchfield, *DERVISH: an office-navigating robot*, AI magazine **16** (1995), no. 2, 53.

[2] R. Simmons and S. Koenig, *Probabilistic robot navigation in partially observable environments*, International Joint Conference on Artificial Intelligence, vol. 14, 1995, pp. 1080–1087.

[3] C.J.C.H. Watkins and P. Dayan, *Q-learning*, Machine learning **8** (1992), no. 3, 279–292.