COMP3702 Artificial Intelligence (Semester 2, 2022)
Assignment 2: HEXBOT MDP – **Report Template**

Name: Zhiyu Chen

Student ID: 45714568

Student email: s4571456@student.uq.edu.au

Note: Please edit the name, student ID number and student email to reflect your identity and **do not modify the design or the layout in the assignment template**, including changing the paging.

Question 1 (Complete your full answer to Question 1 on the remainder of page 1)

State space: All the possible combination of robot position, robot orientation, widget centres, widget orientation in a HexBot environment. That is:

S = {{robot position} x {robot orientation} x {widget centres} x {widget orientation}}

Not all the state in S is valid as {widget centres} x {widget orientation} are limited by the obstacles and hazard to only part of the 2D grid environment. Widget centres could never be in some hex grids.

In solution.py, all possible states are exhausted using the bfs search algorithm, and is stored in self.all_states:

self.win_case, self.all_states = self.bfs(self.environment) (line 77)

Action space: {FORWARD, REVERSE, SPIN_LEFT, SPIN_RIGHT} for robot (All possible actions the agent could take.)

this exists in constants.py below:

ROBOT_ACTIONS = [FORWARD, REVERSE, SPIN_LEFT, SPIN_RIGHT]

Transition function: Given the state s, return the next state s' with the action a taken by the agent where s' returned by Transition function might be the same as s because the action might be invalid. Specifically, there are probabilities that agent would double move or drift and some actions might collide with obstacles or hazard.

This exists in environment.py below:

perform_action(self, state, action, seed=None) (line 467)

Reward function components: $R(s, \pi(s), s')$, in HexBot environment, the reward is consisted of

① ACTION_BASE_COST ②ACTION_PUSH_COST ③Collision penalty and hazard penalty

Agent gets different rewards when taking $\pi(s)$ action.

① And ② are in constants.py, ③ is a given information in ex.txt file

All ①②③ are applied by the apply_dynamics function in environment.py. (line 279)

the purpose of a discount factor in MDPs: Discount factor is used to weight the long-term and short-term rewards. Specifically, the more γ is closer to 1, the greater value of reward in future would have. (But future reward would not be prioritized comparing to recent reward as 0<= γ <= 1)

Planning Horizon: indefinite horizon **Sensing Uncertainty:** Fully observable **Effect Uncertainty:** stochastic

Computational Limits: bounded rationality **Learning:** knowledge is given

Question 2 (Complete your full answer to Question 2 on page 2)

a) For VI, each iteration loop through all states and evaluate best actions to take based on value of all possible actions taken by agent in this state.
For PI, firstly generate a converged value for each state using random actions taken by agent in this state, then evaluate best actions to take in this state until the policies are converged.

I did not use asynchronous for both.

Policy evaluation update the policy(action taken) in each state by evaluating all possible actions using the existing value of each state and choose the one with highest value.

b)

Test cases	VI		PI	
	Time to converge	Number of iterations	Time to converge	Number of iterations
Ex1.txt	4.91	49	5.19	50
Ex2.txt	10.61	70	10.84	75
Ex4.txt	17.71	169	14.08	132

c) For complicate cases, PI would have better performance, the reasons are:

1. The calculation is linear in PI as the max() function to choose the best action is not iterated in the process of reaching converge value.
2. If number of states are small, PI need to perform more iteration times to reach the converge value for each policy. While VI only need to reach converge value once, it could be a litter faster in these cases.

Question 3 (Complete your full answer to Question 3 on page 3)

a) Both hazard penalty and transition probabilities contribute to the value of each state and then the best action evaluated to take. Specifically, the higher penalty applied, the less likely that agent would take the risk of the top half route. The higher transition probability, the risky for the top half route, hence less likely for agent to choose the top half route.

b) Take hazard to 0, robot would very like to choose shorter way to destination as the value of states in top cells would be higher without penalty.

Take hazard to ∞(float("inf")), robot would completely avoid any possibility to collide with hazard, so that all actions for top half cells leads the way to bottom half cells.

Take probability = 0 for any action, collide to hazard would never happen so the top half closer route would be chosen.

Take probability = 0.5 for drift or double move, the result of transition functions would be unpredictable, so the agent would choose safer bottom part route to avoid penalty.

In short, both hazard penalty and transition probabilities are factors contributing to value of each state. The determining factor is still value of each state to evaluate actions.