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

Name:

Student ID:

Student email:

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)

a) The state space is every valid combination of widget locations and robot locations and orients. The function that produces a list of all these possible states can be found at line 289 in solution.py in the get_states method. The action space is the list of possible actions that the robot can do or could end up doing, including drift and double moves. The actions themselves can be found in the constants.py file, however the double moves and drifts and their probabilities can be found at line 255 of solution.py in the stoch_actions method. The transition function and reward functions were combined in the perform_action method at line 458 of environment.py and this function applies the probabilities of the action actually happening or the drift or double move happening, then returns the new state as a result of these probabilities and then the reward/cost for being in this new state

b) The discount factor is used partially as a way to make the infinite sum finite and thus computable, however it is also useful as the decision maker is uncertain about whether the next instance of the state is actually solved or not. This is because we don't know if the solving move will be performed or not.

c)

- Planning Horizon: Indefinite horizon as it is unknown when the environment will enter a solved state. We can't just stop when we think we have done a finishing move as there is the chance that the move does not happen. This means that we have to keep going on potentially forever only stopping once we arrive at the solved state
- Sensing Uncertainty: Fully observable as the agent always knows exactly what is around it at all times and can see every state and see the state it is currently in
- Effect Uncertainty: stochastic as there is a probability of each action happening or there being drift or a double move occurring or both.
- Computational Limits: perfect rationality, this is because there is not a timed factor to this environment and so the agent will just commit the actions as soon as it knows what it should do, without regard for the computing power required to know what action it should do
- Learning: given, this is because the agent is given all of the knowledge about probabilities and what the costs/rewards of being in each states gives and will make an action based on the information given

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

a) .

   i. value iteration was implemented in-place and asynchronously.

   ii. Policy evaluation was completed using linear algebra and

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

a) For the rest of this question I will refer to the transition probabilities increasing as the chance of there being no drift or double move increasing. You would expect with an increase in hazard penalty or a decrease in probability, that the robot would avoid the high risk path and go for the easier path down below, however if you adjusted either of these too much in this direction I would expect that the algorithm would simply spin and never find the goal as the paths are all too high cost to attempt and so the best solution would be to just spin on the spot. And then as you increase the probabilities and decrease the hazard cost the robot would tend towards the higher risk path along the top as there is a decent chance that the high cost doesn't happen and so the reward is considered much higher.

b) Hazard penalty increase to 15, decrease to 5.

probabilities increase to 0.95 (double_move_forward = 0.2, drift_probs_forward = 0.025),

(double_move_reverse = 0.05, drift_probs_reverse = 0.0125)

probabilities decrease to 0.26 (double_move_forward = 0.3, drift_probs_forward = 0.1),

(double_move_reverse = 0.25, drift_probs_reverse = 0.05)

i. Hazard penalty increase, probabilities increase: top path (high risk, high reward)

ii. hazard penalty increase, probabilities decreased: continues to attempt top path but fails

iii. hazard penalty increase, probabilities remain the same: top path (high risk, high reward)

iv. hazard penalty decrease, probabilities increase: top path (high risk, high reward)

v. hazard penalty decrease, probabilities decrease: top path (high risk, high reward)

vi. hazard penalty decrease, probabilities remain the same: top path (high risk, high reward)

vii. hazard penalty remains same, probabilities increase: top path (high risk, high reward)

viii. hazard penalty remains same, probabilities decrease: top path (high risk, high reward)

ix. hazard penalty remains same, probabilities remain the same: top path (high risk, high reward)

It appears that nothing can deter the agent from attempting the top path, I think this has to do with how much shorter the path is compared to the top path, and so it seems like no amount of hazard penalties will deter it, even attempting up to 50 as the penalty alongside the decreased move probabilities. I think this could have something to do with my implementation using vi instead of pi.