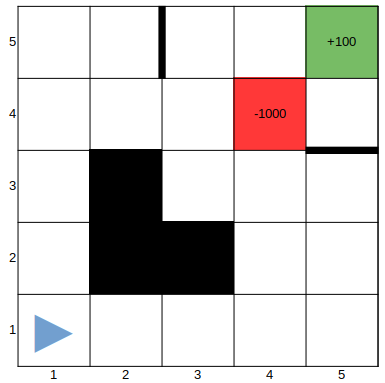
# **CPSC 4420/6420: Artificial intelligence**

# Assignment 2

# Name:

Consider the following puzzle. The green and red states are both terminal states, with the rewards as shown (so we can consider the green state the “goal”, and the red a “game over” state with a large negative reward). Thick borders between cells represent walls that the robot player cannot cross, and the black squares contain obstacles and cannot be entered. The robot player is represented by the blue triangle, and the direction the triangle points is the way the robot is facing.



The robot can take the following actions: 10.5 10

A1: Move one cell forward in the direction it is facing. Cost: 1.5

A2: Move two cells forward in the direction it is facing. Cost: 2

A3: Turn to its left, and stay in the same cell. Cost: 0.5

A4: Turn to its right, and stay in the same cell. Cost: 0.5

Note that each action has a cost. This can also be considered an immediate negative reward

(so R(s,A1,s’) = -1.5). The cost is evaluated on the current state, the one the robot is in when it begins the action, not the one it ends up in after the action. The same way is the value of state V(s) shows the value of current state and you should initialize the algorithm with V1(5,5,x)=+100, V1(4,4,x)=-1000 (for x=1,2,3,4), and zero for all other states.

Let’s represent a state with (x,y,d), where x and y represent horizontal and vertical position (i.e. location), and d represents the direction the robot is facing (1: up, 2: down, 3: left, and 4: right).

So, for example, if the robot is in state (4,1,4), it means that it is in location (4,1) and facing right. The result of possible actions for this state are as follows:

1 forward 2 forward2 3 turn right 4 turn left

A1 (move 1 cell forward) --> (5,1,4)

A2 (move 2 cells forward) --> impossible

A3 (turn left) --> (4,1,1) : the robot stays in (4,1) but now faces up

A4 (turn right) --> (4,1,2) : the robot stays in (4,1) but now faces down

A move is impossible if it would result in landing on blocked cells (2,2), (2,3), or (3,2), or if it would result in crossing a barrier, like moving from state (2,5) to (3,5), or (5,3) to (5,4). A move that would take the robot outside of our 5x5 grid is also impossible.

Note that we have more states than there are cells, because the robot facing a different direction produces a new state, even if it does not change location. In the example above, if we move to (4,1,1), where the robot is facing up, this is a different state from the one we were in, (4,1,4), even though the robot has not moved cells.

1. If there is no living reward, no noise, and no discount (gamma = 1), use your common sense to find the best possible route from (1,1) to (5,5).

Using common sense I calculated the best route using the fewest costs to be -10, this is by going straight to (1,4,4) then going up to (4,5,1) and turning right to the target goal. Going the other way, results in -10.5 which has a higher cost of 0.5.

1. With no discount (gamma = 1), no living reward, and no noise, use the Value Iteration Algorithm with 100 iterations to update the optimal values for each state and print the result [only for the first 10 iterations] in the following format:

iter 1:

state (1,1,1) V = (some value) Best Action: Ai (where i is some number 1-4)

state (1,1,2) V = (some value) Best Action: Aj

…

state (5,5,4) V = (some value)

iter 2:

state (1,1,1) V = (some value) Best Action: Ai (where i is some number 1-4)

state (1,1,2) V = (some value) Best Action: Aj

…

state (5,5,4) V = (some value)

If two actions are tied for best, you can select one at random or always choose the one with the smallest index.

Select a from the options to run this

1. If you start from state (1,1,4) and follow the optimal policy you found in part B, does it follow the same path you proposed in part A?

No, from this position, your best path is to go right passing the blocked cells. Then you go up to the cell below the negative reward and then jump over it and turn right to get to finished terminal cell.

1. Repeat part B with the same assumptions, except for gamma = 0.8. Compare the results with that from part B. Do they match?

Yes, the results matches the results from B for the best optimal path but the Values are very different.

1. Repeat part B with the same assumptions, except for gamma = 0.2. Compare the results with that from parts B and D. Do they match?

No the results are not the same because the Values are too low (0) and most of the policies for the best actions are still defined with the initial random policy.

1. **(Optional for 4420)** Repeat part B, but this time with noise = 0.2, and gamma = 0.9 and no living reward. With a noise of 0.2, every time you take an action, the result will be the expected action with probability 0.8 (80%), but 20% of the time, the robot will instead take a different action (taken randomly out of possible unexpected actions, with equal probability).

For example, if we are in state (4,1,4), so location (4,1) and facing right, and we take action A1 (moving one cell forward), the result will be:

A1: (5,1,2) with probability 0.8

A2 is impossible

A3: (4,1,1) with probability 0.1

A4: (4,1,2) with probability 0.1

Compare the results with that of the previous parts and explain your observations.