Solutions for HW2

Yunhai Han
Department of Mechanical and Aerospace Engineering
University of California, San Diego
y8han@eng.ucsd.edu

April 27, 2020

Contents

1 Problem 1

1.1 problem formulation

(Drunkard's walk, 15 points). A man walks along a four-block stretch of Park Avenue. If he is at corner $1, 2$, or 3, then he walks to the left or right with some probability. Assume that the probability of a step to the right is $2/3$, and to the left is $1/3$. He continues until he reaches corner 4 which is a bar, or corner 0, which is his home. If he reaches either home or the bar, he stays there. We form a Markov chain with states 0, 1, 2, 3, and 4. Is the Markov chain absorbing? Write the transition matrix in a block diagonal form, finding Q and R. Obtain $N, N1$ and NR. Interpret the results.

1.2 solution

The transition matrix could be represented as:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

You could see there are two absorbing states:state 0 and state 4. Because if he reaches these two states, he would stay there.

For a absorbing Markov chains, there are two requirements:

- There is at lease one absorbing state
- There is a path connecting any node to an absorbing state in the MC transition graph

From any node(1,2,3), there is a path connecting them to the absorbing state(0 or 4). Hence, it is absorbing.

In order to write the transition matrix in a block diagonal form, I have to switch the order of the original states. Hence, the new order is:1,2,3,0,4. The new transition matrix P is:

$$P = \begin{bmatrix} 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{2}{3} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

And this is the canonical and fundamental matrix of an absorbing Markov chain

$$P = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix} \quad N = (I - Q)^{-1}$$

Q is TR-TR state block, R is TR-ABS block, I is ABS-ABS block.

$$Q = \begin{bmatrix} 0 & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & \frac{1}{3} & 0 \end{bmatrix} R = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & 0 \\ 0 & \frac{2}{3} \end{bmatrix} N = \begin{bmatrix} 1.4 & 1.2 & 0.8 \\ 0.6 & 1.8 & 1.2 \\ 0.2 & 0.6 & 1.4 \end{bmatrix}$$

Each element n_{ij} in the matrix N is the expected number of times in transient state j given it started in the transient state i.

Each element in NR represents the probability that an absorbing MC will be absorbed by j if it starts at transient state i. Each element in $N1$ represents the expected number of steps before the chain is absorbed.

They could be obtained respectively as follow:

NR : Suppose b_{ij} is the probability that an absorbing MC will be absorbed by j if it starts at transient state i. We could write this system of equations:

$$\begin{cases} b_{10} = \frac{1}{3} + \frac{2}{3}b_{20} \\ b_{20} = \frac{1}{3}b_{10} + \frac{2}{3}b_{30} \\ b_{30} = \frac{1}{3}b_{20} \\ b_{14} = \frac{2}{3}b_{24} \\ b_{24} = \frac{1}{3}b_{14} + \frac{2}{3}b_{34} \\ b_{34} = \frac{2}{3} + \frac{1}{3}b_{24} \end{cases}$$

By solving this linear equations, we could obtain the matrix NR.

$$NR = \begin{bmatrix} \frac{7}{15} & \frac{8}{15} \\ \frac{1}{5} & \frac{4}{5} \\ \frac{1}{15} & \frac{14}{15} \end{bmatrix}$$

The sum of each row is equal to 1 and this means it will always be absorbed from any transient states.

$N1$: Suppose t_i is the expected number of steps before the chain is absorbed when the chain starts in state i.

There are two absorbing states in this case, so I first reconstruct the graph so that there is only one absorbing state. This could be done by simply combining the two absorbing states into one. And the new transition matrix P is:

$$P = \begin{bmatrix} 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then I could write down a system of equations:

$$\begin{cases} t_1 = 1 + \frac{1}{3} * 0 + \frac{2}{3}t_2 \\ t_2 = 1 + \frac{1}{3}t_1 + \frac{2}{3}t_2 \\ t_3 = 1 + \frac{1}{3}t_2 + \frac{2}{3} * 0 \end{cases}$$

By solving this linear equations, we could obtain $N1$:

$$N1 = \begin{bmatrix} \frac{57}{17} \\ \frac{60}{17} \\ \frac{36}{17} \end{bmatrix}$$

2 Problem2

2.1 problem formulation

Consider the parametric genetic MC model in the slides. Complete the model by finding the parameter-dependent transition matrix, $P(gg)$, corresponding to one of the parents being recessive gg. Which of the $P(GG), P(Gg)$ or $P(gg)$ is absorbing? Suppose we mate successive individuals, so that we start mating one hybrid individual with a dominant GG individual, and then their offspring with a hybrid individual, and successively, the offspring of couple n with a hybrid individual. What is the individual type as $n \to \infty$? Justify the result.

2.2 solution

Suppose one of the parents is gg, we could compute the transition matrix $P(gg)$:

Other parent is \ Offspring is	GG	Gg	gg
GG	0	1	0
Gg	0	0.5	0.5
gg	0	0	1

Hence the matrix $P(gg)$ is:

$$P(gg) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Since p_{gg2gg} is one, it is an absorbing state. Besides, we could always find a path connecting from any nodes to this absorbing state. So $P(gg)$ is absorbing. The transition matrix of other two models are:

$$P(GG) = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 1 & 0 \end{bmatrix} P(Gg) = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

$P(GG)$ is also absorbing because p_{GG2GG} is one.

We start mating one hybrid individual with a dominant individual, so their offspring maybe hybrid or dominant with equal probability 0.5 for each. Besides, all the successive offspring will mate with a hybrid individual, so the transition matrix is:

$$P(Gg) = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

We could multiple $bel(x_0)$ by $P(Gg)$ many times and you could see the results would converge.

$$P(Gg)^{100} = P(Gg)^{101} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.5 & 0.25 \end{bmatrix}$$

Hence $bel(x_t) = (0.25, 0.5, 0.25)$ as $n \to \infty$.

3 Problem3

3.1 problem formulation

In this exercise, you will implement and demonstrate the performance of stochastic search algorithms in different environments. To do this, please download the folder "mdps" from canvas. In it you will find several updated files.

a Edit valueIteration.py, to use smal1Grid. Implement the (backward) value iteration policy, returning J_0 in the valueIteration function. The iterations should use the discount parameter γ and the cost function associated with the WINSTATE, LOSESTATE, defined in getcost. In addition to the value iteration function, implement 2 functions more: (a) getpolicy(J), which returns the best control action according to a vector of J cost-to-go values, and (b) getqvalues (x, u, Q), that calculates the $Q(x, u)$ value from a previous table Q. You can use these functions inside the value iteration. For what value of T does it seems that value iteration converges with $\gamma = 0.9$ and $\eta = 0.2$?

b Using mediumGrid and the standard getcost function, play with the discount factor γ and the default noise of η in the Markov chain, to obtain different results in this environment with getcost function. Can your optimal policy crosses through the narrow bridge given by the obstacles in the environment?

c (15 points) Consider the mediumGrid layout and the getcostBridge function. This grid has two terminal states with positive payoff at (5,3) and at (3,3) . The bottom row of the environment have states with negative payoff - 10 (cliff states). The starting state is at (1,2). We distinguish between two types of paths: (1) paths that "risk the cliff" and travel near the bottom row of the grid; these paths are shorter but risk earning a large negative payoff. (2) Paths that "avoid the cliff" and travel along the top edge of the grid. These paths are longer but are less likely to incur huge negative payoffs (or large costs.)

For each of the following optimal policy types (1) through (4), give an assignment of parameter values for discount, and altering the noise which produce the policy or state such that: 1) Prefer the close exit (with cost -1), risking the cliff (with cost +10) 2) Prefer the close exit (with cost -1), but avoiding the cliff (with cost +10) 3) Prefer the distant exit (with cost -10), risking the cliff (with cost -10) 4) Avoid both exits (also avoiding the cliff)

3.2 solutions

3.2.1 a

As required in the problem, I use **smallGrid** and edit the functions *valueIteration*,*valueIteration*,*getPolicy* and *getQvalues*. Here are the results with $\gamma = 0.9$ and $\eta = 0.2$.

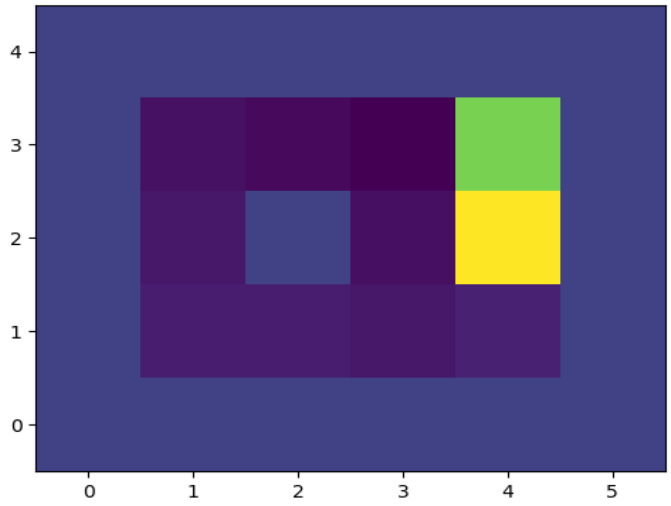

Figure 1: Time horizon T=30

In the above figure, the yellow cell corresponds to **LOSESTATE** and the green one corresponds to **WINSTATE**. I tune the parameter T as follow:

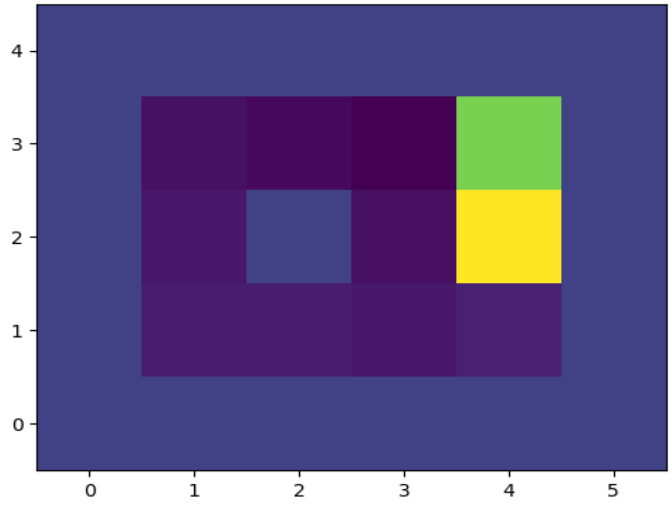

Figure 2: Time horizon T=20

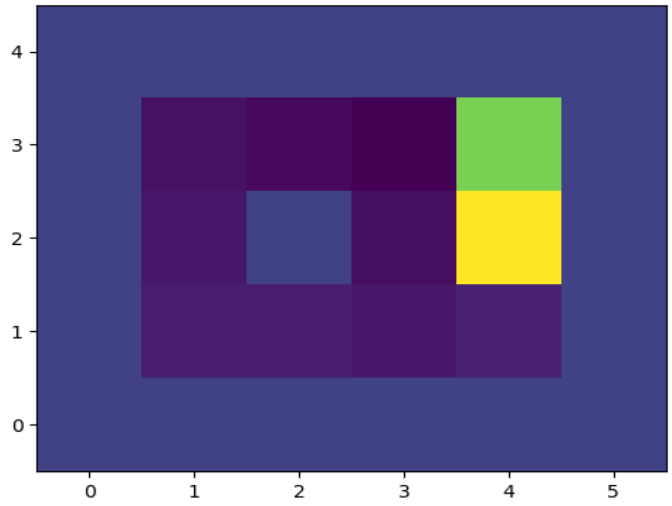

Figure 3: Time horizon T=10

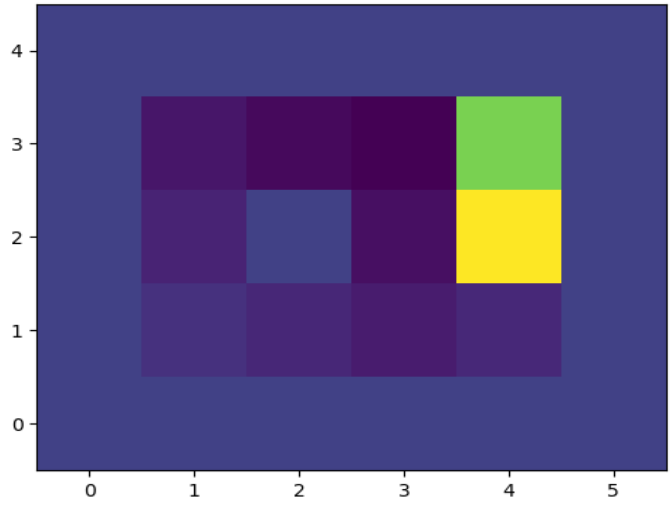

Figure 4: Time horizon T=5

There seems no differences between the final results of $T = 10, T = 20$ and $T = 30$ and the differences between the results of $T = 10$ and $T = 5$ are clear. Hence, the value iteration seems to converge if T is larger than 10.

The optimal path(drawn in pink) from the start(1,1) to the goal(4,3) is shown as follow:

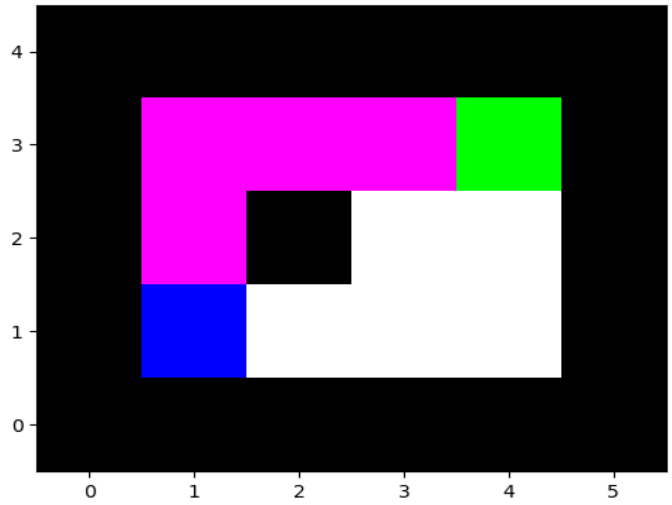

Figure 5: The optimal path

Also, I verify the validity of the two functions:

```
valueIteration(5,0.9)
print(getQvalues([1,2],(0,1),5,0.9))
print(getPolicy([1,2], 5))
```

Figure 6: Codes

```
-0.4598415360000002
(0, 1)
```

Figure 7: print contents($T = 5$)

The optimal control action at $x = [1, 2]$ is $u = (0, 1)$ and the q-value of the state-control pair ([1,2],(0,1)) is -0.46. This value is the same as it I obtained from the grid graph. These results are obtained when T is set as 5.

```
-0.63412775924107
(0, 1)
```

Figure 8: print contents($T = 10$)

```
-0.6371143394376754
(0, 1)
```

Figure 9: print contents($T = 20$)

```
-0.6371146985515909
(0, 1)
```

Figure 10: print contents($T = 30$)

3.2.2 b

First, I set $\gamma = 0.9$ and $\eta = 0.2$:

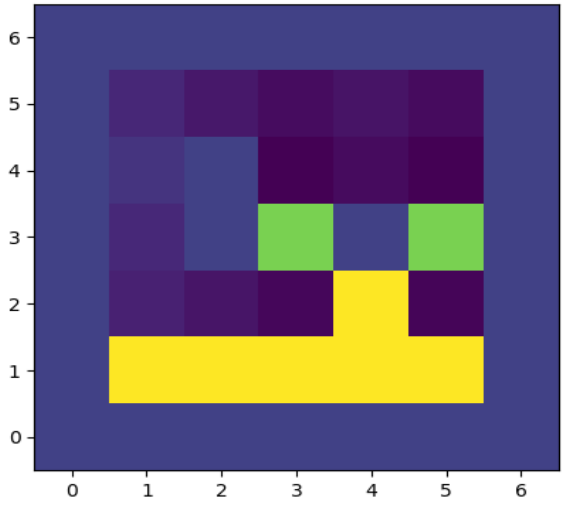

Figure 11: Time horizon T=5

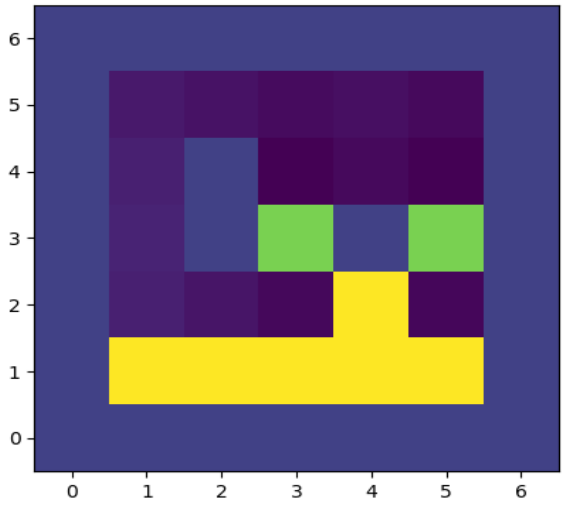

Figure 12: Time horizon T=10

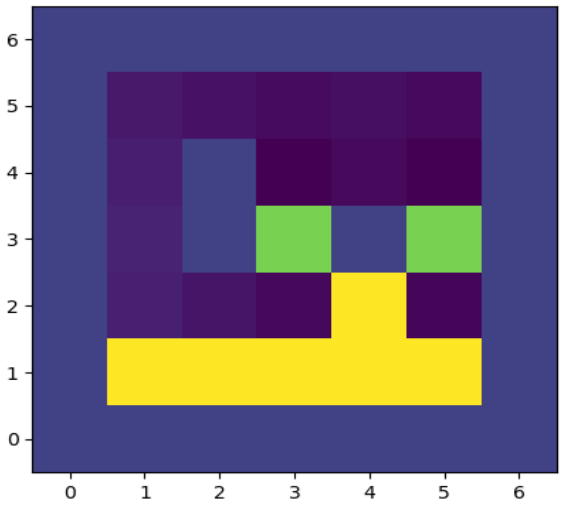

Figure 13: Time horizon T=20

Then, I set $\gamma = 0.5$ and $\eta = 0.2$:

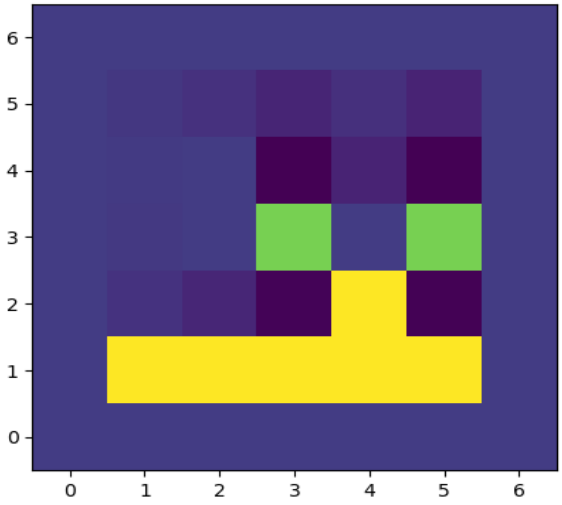

Figure 14: Time horizon T=5

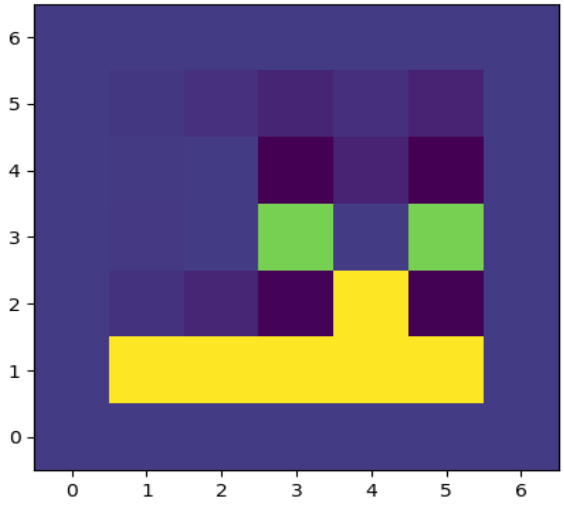

Figure 15: Time horizon T=10

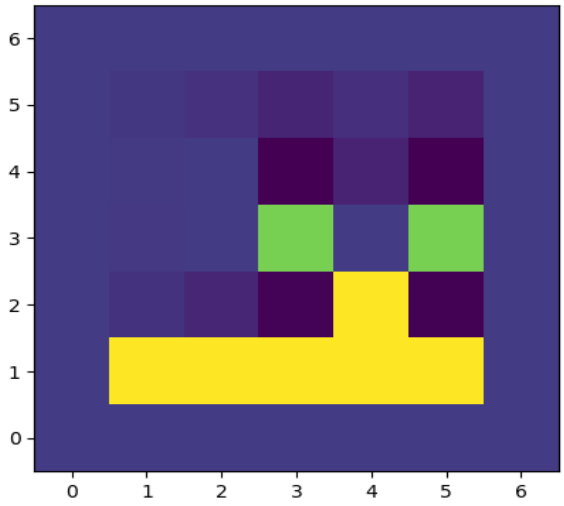

Figure 16: Time horizon T=20

Finally, I set $\gamma = 0.1$ and $\eta = 0.2$:

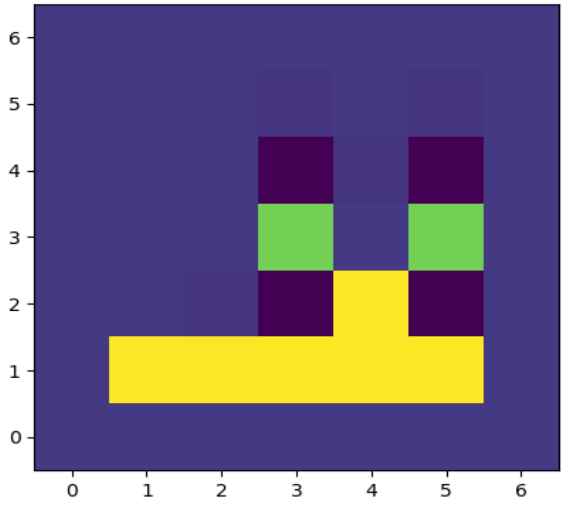

Figure 17: Time horizon T=5

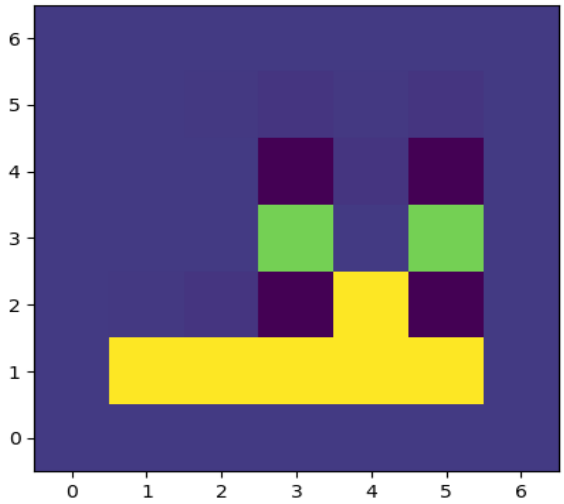

Figure 18: Time horizon T=10

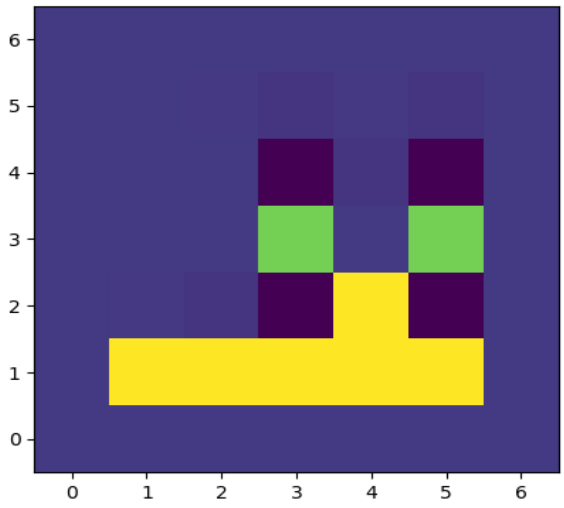

Figure 19: Time horizon T=20

As $gamma$ decreases, during each iteration, it is harder for these cells which are far away from the goal node to find the optimal path. This statement could be proved by the larger penalty value corresponding to each cell. This is because this algorithm tends to be greedy instead of exhaustive but no greedy move could lead them to the final goal. This can be dangerous for the cells near the narrow bridge especially when the motion noise grows. In this case, with $\eta = 0.2$, even I set $\gamma = 0.01$, the algorithm also finds a optimal path from the start(1,2) to the exit(3,3). However, the q-value at the start is almost zero(-5.16e^{-5})

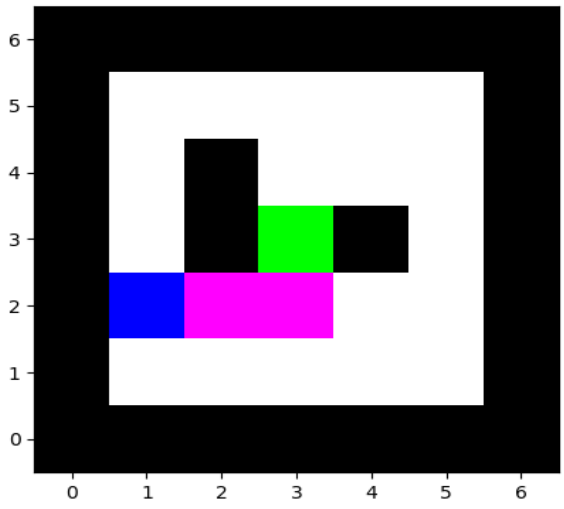

Figure 20: The optimal path

The q-value at the node(1,4) is smallest among all the node(-3.3e^{-9}), but it could also finds the optimal path:

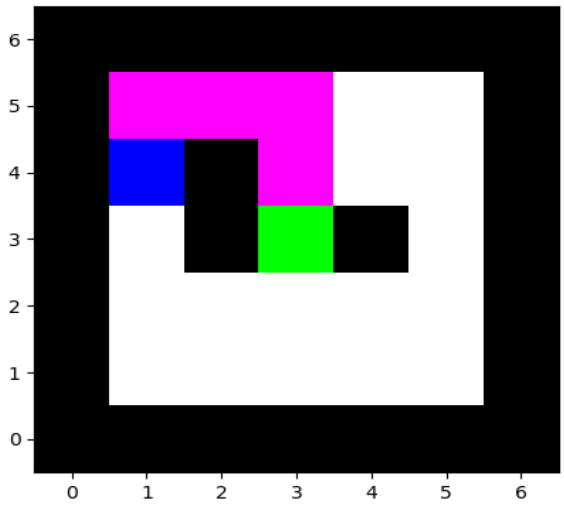

Figure 21: The optimal path

With the default η, the algorithm still works but it becomes less robust if we decrease γ.

3.2.3 c

1 First, I try the previous parameter values:$\gamma = 0.9$ and $\eta = 0.2$.

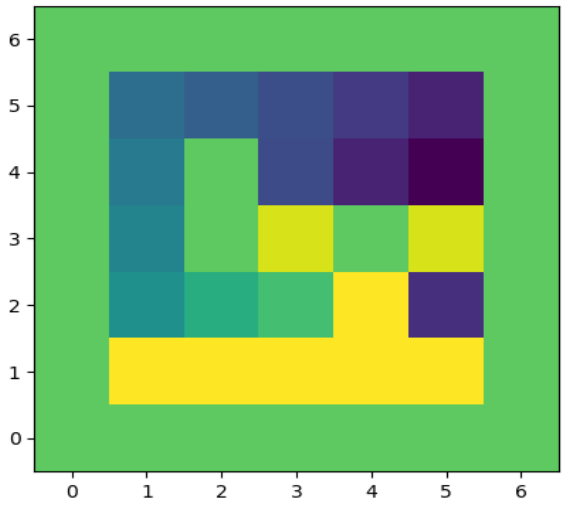

Figure 22: Time horizon T=20

However, since the distance exit has a much lower cost, the algorithm prefers this exit instead of the close one. Even starting from (2,2), the optimal path still connects it with the distant exit.

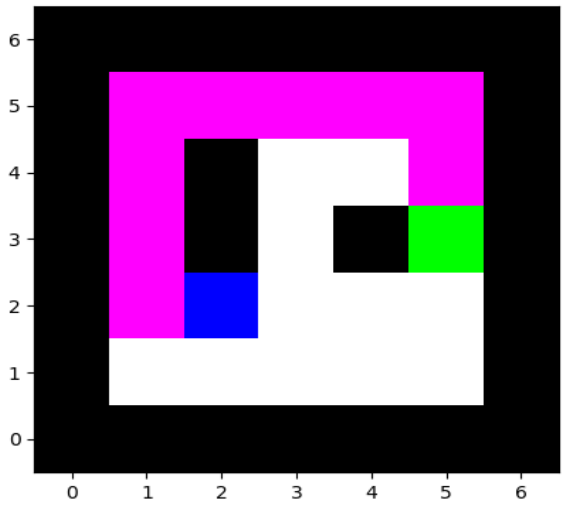

Figure 23: The optimal path starting from (2,2)

By tuning the parameter values, we are able to let it prefer the close exit and risk the cliff(travel near the bottom row of the grid). In order to select the shorter path with much larger negative payoff, the motion noise is very important. Hence, I set $\eta = 0$, which means there is no motion noise. On the other hand, in order to choose the close exit instead of distance exit with lower cost, γ should also be decreased, which makes the algorithm less exhaustive to find the real global optimal path.

Hence, the new parameter values are $\gamma = 0.3$ and $\eta = 0$.

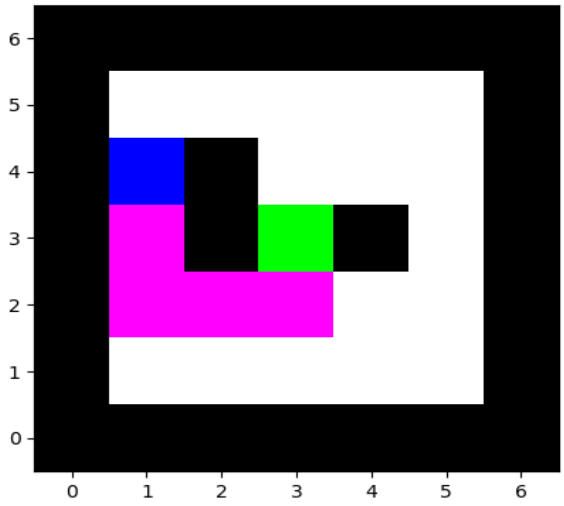

Figure 24: The optimal path starting from (1,4)

From the above figure, starting from (1,4), the algorithm could give an optimal path travelling near the bottom row.

2 By tuning the parameter values, we are also able to let it prefer the close exit but avoid the cliff. If the motion noise exists, when we travel near the bottom row, larger cost values are

possible to be added into the Q-value and this could make the algorithm choose a path that avoids the cliff.

Hence, the new parameter values are $\gamma = 0.3$ and $\eta = 0.1$.

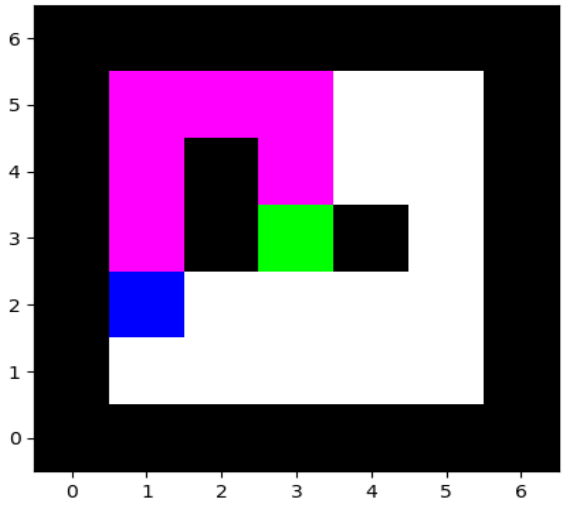

Figure 25: The optimal path starting from (1,2)

3 In order to make the algorithm prefer the distant exit and risk the cliff, the motion noise should be small(decrease η) and γ should be large, which makes it more exhaustive.

Hence, the new parameter values are $\gamma = 0.9$ and $\eta = 0.1$.

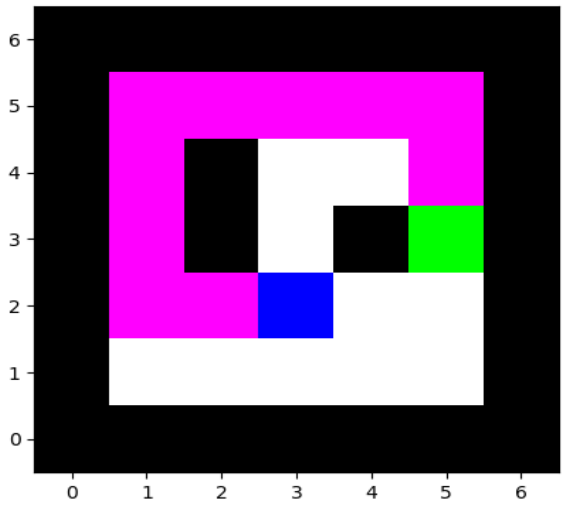

Figure 26: The optimal path starting from (3,2)

Even starting from (3,2), it would try to reach the distant exit instead of the close one, which is only one cell away from the start node.

4 In order to avoid both exits and the cliff, we have to increase η, which makes it more possible to earn large negative payoff. So, I set $\eta = 0.4$.

There is a special case, no matte how I tune γ, the optimal action for the starting node(2,2) is always not moving. The reason is that if it moves right, it could get closer to the close exit but also has a chance to take a large negative payoff due to the existence of the motion noise. If it moves left, the same thing happens. Hence, the best action is to move up, encountering the obstacle and then move back.

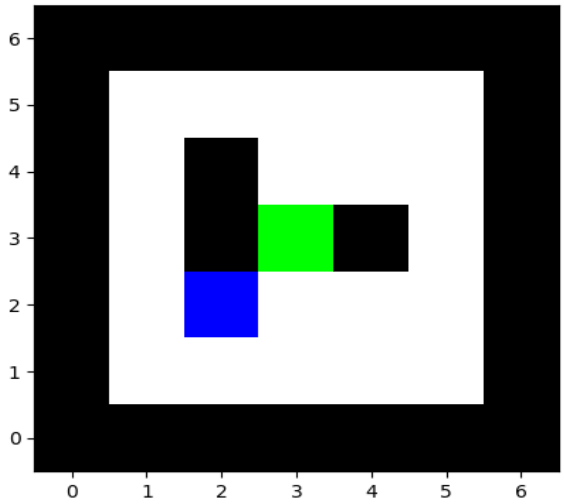

Figure 27: The optimal path starting from (2,2)

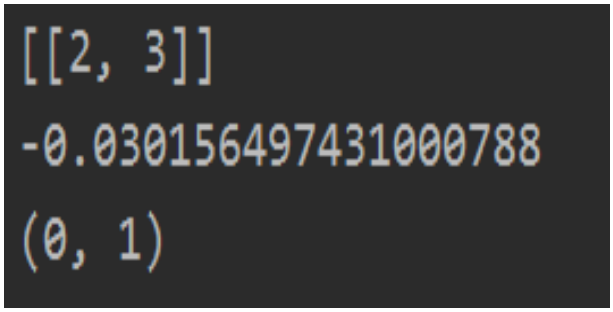

Figure 28: Terminal outputs

You could see the best action is (0,1)(move up).

Besides, if I set $\gamma = 0.2$ and $\eta = 0.4$ and the start node is (1,2), the best action is (0,1)(move up). This is because the algorithm prefers the close exit(γ is small) and avoids the cliff(η is large). However, the best action of the node (1,3) is (0,1)(move down). This is because (1,3) is not directly connected to any cliff cell, so even if it chooses the shorter path along the bottom row, the large negative payoff will not be added into its Q-value(unreachable). Hence, if we start at (1,2) or (1,3) we would never get out of these two nodes: the best actions would alternate between (0,1) and (0,-1).

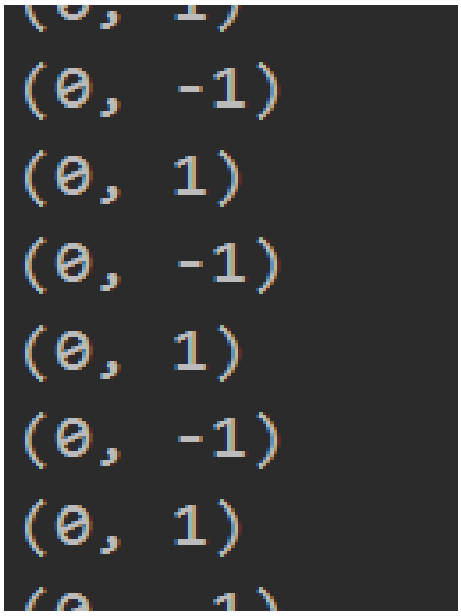

Figure 29: Terminal outputs