CS312 Lab 8 Report

Anudeep Tubati, 170010039

I. OVERVIEW

This report elaborates on the Policy Iteration and Value Iteration methods used to obtain optimal policy for a particular (Total Reward) Markov Decision Process and the results obtained from varying the process' parameters.

II. DESCRIPTION OF MDP

a. State Space (S)

Environment is a 2D graph of NxN square tiles. Each tile can be a free tile (denoted by space ' '), a tile with the agent on it (denoted by 'A'), a tile with a detector plate (denoted by 'D') or a goal tile (denoted by 'G').

The agent starts at a *start_position* always. The agent has to reach the goal tile. Upon reaching the goal tile, the agent exits the game and is granted a reward *final_reward*.

b. Actions (A)

The agent's colleague at the base radios the next action to the agent. The possible actions are [Up, Down, Left, Right]. If the agent is not able to perform the action sent on radio, she stays in the current tile itself.

c. Transition Probabilities (P)

Since wireless communication involves some noise, there is *p_err* probability of an action sent on radio being received as one of the remaining 3 actions. For eg., if Up is sent, there is a probability that Up is received as Up with (1 - *p_err*) probability and as the rest 3 actions with *p_err*/3 probabilities each.

The agent executes the action finally received by her.

The table with transition probabilities is given on the next page.

	Received				
Sent		**Up**	**Down**	**Left**	**Right**
	Up	(1 - *p_err*)	*p_err*/3	*p_err*/3	*p_err*/3
	Down	*p_err*/3	(1 - *p_err*)	*p_err*/3	*p_err*/3
	Left	*p_err*/3	*p_err*/3	(1 - *p_err*)	*p_err*/3
	Right	*p_err*/3	*p_err*/3	*p_err*/3	(1 - *p_err*)

d. Rewards (R)

There are only 3 rewards in the environment.

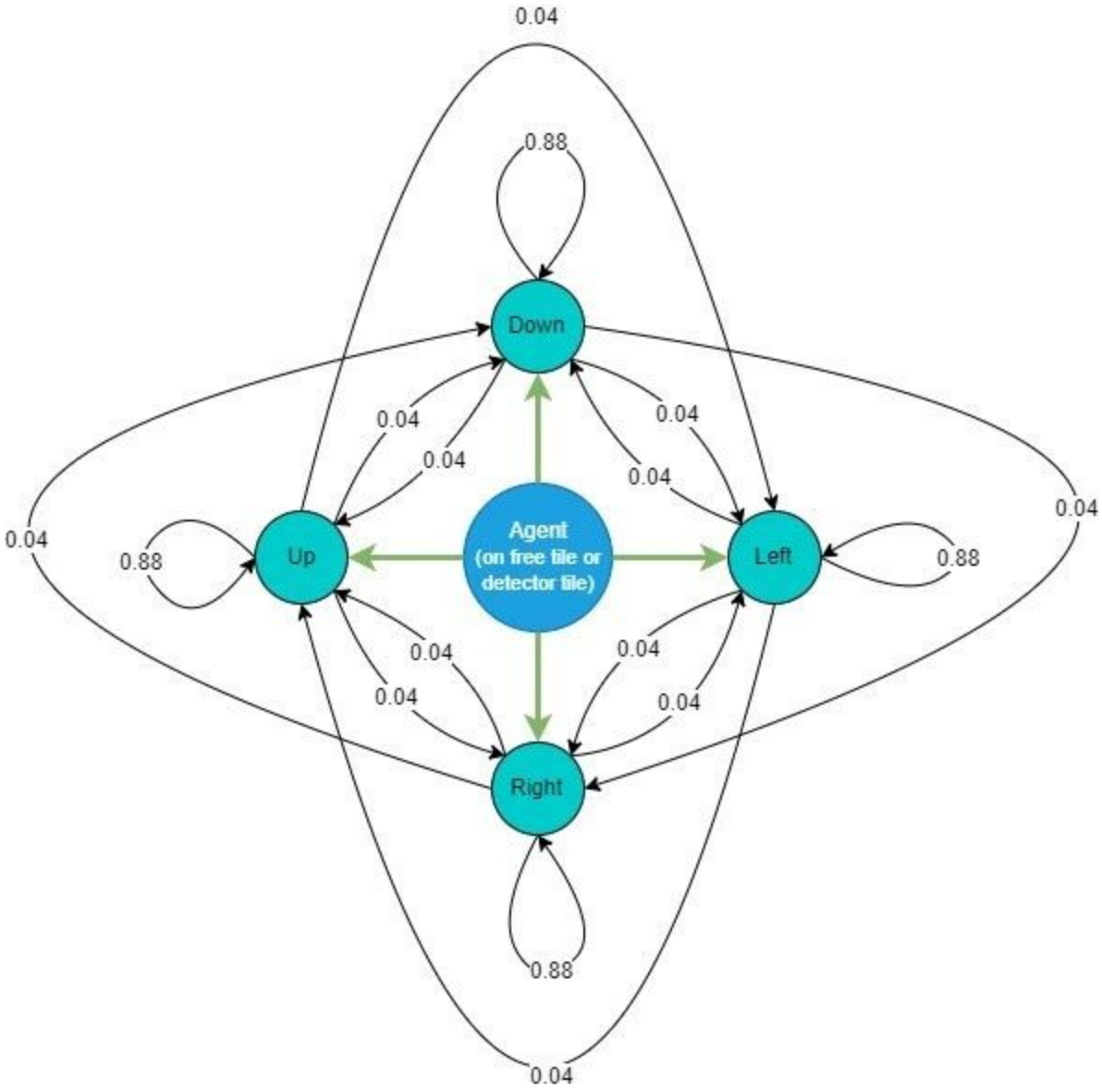
1. Executing an action (*-step_pen*) - Executing any action gives the agent a penalty of *delta*. This is to prohibit the agent from staying in a tile indefinitely, as it is a Total Reward process.
2. Stepping on a detector plate (*-d_pen*) - Stepping on a detector plate penalises the agent by *d_pen*.
3. Reaching goal (*+fin_rew*) - Infiltrating the lab successfully (reaching the goal) rewards the agent with *+fin_rew* and exits the game.

e. Horizon (N)

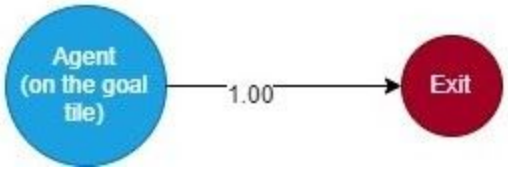
As this MDP has a non-zero chance of error (i.e., Up changing to Down, Left or Right) on each step, there is some non-zero randomness in the agent's path. This ensures there is a non-zero probability of the agent reaching the Goal (Exit State) irrespective of the actions sent to her. Therefore, we do not need to constraint the MDP to a certain depth (Horizon) as it is always guaranteed to reach an Exit State.

III. STATE-TRANSITION GRAPH

p_err = 0.12 is used for the Graph



Green arrows represent the action sent on radio. After that, the transition for action into other actions are shown with the black arrows



When the Agent reaches the goal, she gets a reward of *fin_rew* and exits

IV. OPTIMAL POLICY

To verify the policy obtained by the aforementioned methods, we can give 0 rewards for every state except a test state. The obtained policy always takes us to the fixed test state only.

As we can conclude from the above images, the policy always leads every tile to the fixed test state (G) in the **shortest possible steps**. Hence, the obtained policy is optimal.

V. EXPERIMENTAL RESULTS

Since it is a Total Reward MDP (not a Discounted Reward MDP), we fiddle with the *step_pen* value, not *gamma*.

The environment to be used for this experiment is

4x4 tiles; start_position = (0, 2); Detector plate at (1, 2) and Goal at (3, 2)

step_pen	Number of Iterations to converge	
	Value Iteration (VI)	**Policy Iteration (PI)**
-0.01	7	6
-0.05	7	5
-0.1	7	6
-0.3	7	7

As we can infer from the table, the convergence **does not** depend on the *step_pen* value.

The obtained policy from both the methods has negligible difference.

Case I - *step_pen* significantly lesser than *d_pen*. Also, *p_err* is a small constant.

step_pen = -0.01, *d_pen* = -0.5, *p_err* = 0.12

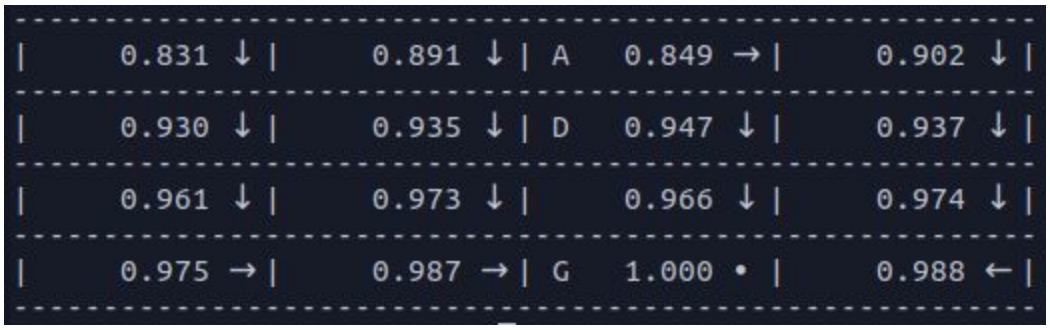


Fig a. Policy Iteration

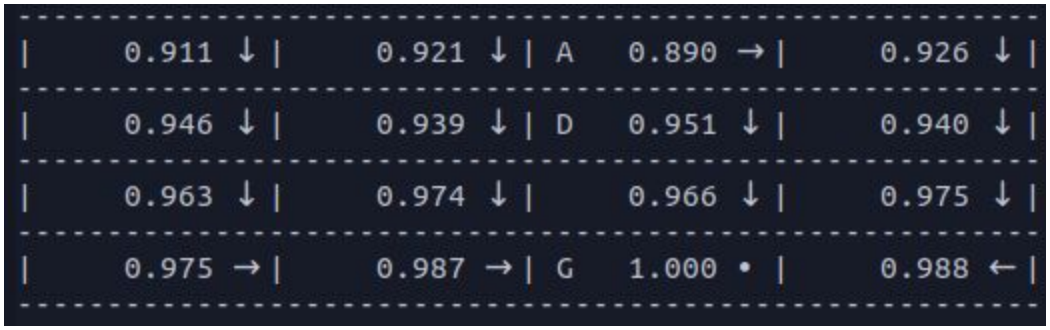


Fig b. Vaule Iteration

As it can be seen from the image, the policy tries its best to avoid the detector plate and take a longer route. This is because the penalty incurred by going over detector plate is greater (0.5) than the penalty incurred by taking 2 extra steps than optimal tour (0.02).

Case II - *step_pen* almost equal to or greater than *d_pen*. Also, *p_err* is a small constant.

step_pen = -0.3, *d_pen* = -0.5, *p_err* = 0.12

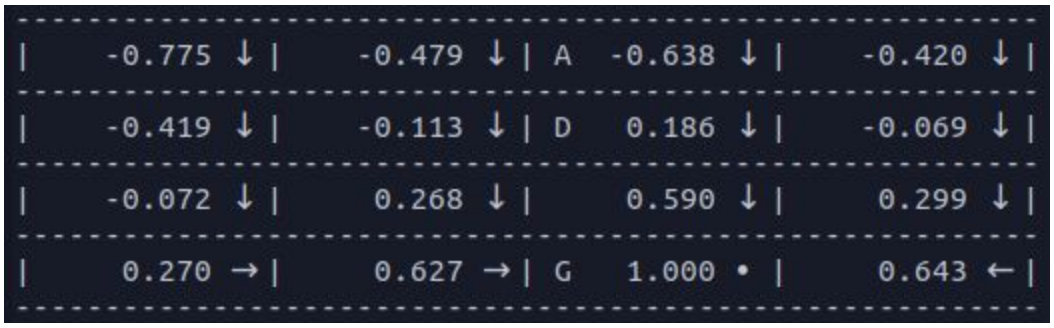


Fig c. Policy Iteration

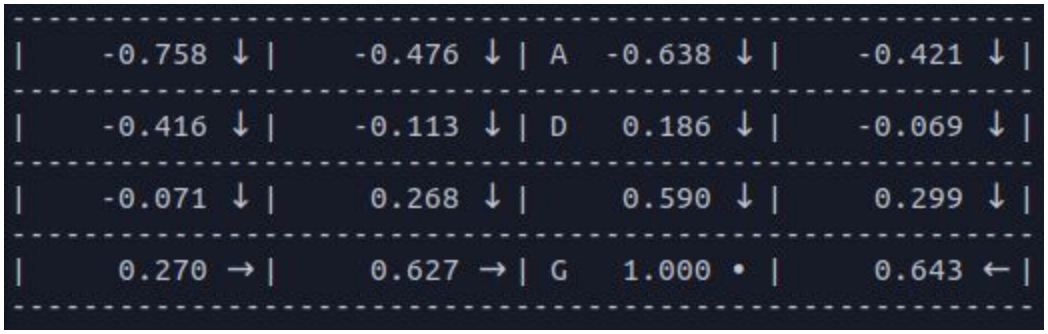


Fig d. Value Iteration

As it can be seen from the image, the policy goes over the detector plate, straight to the goal. This is because the penalty incurred by going over detector plate is lesser (0.5) than the penalty incurred by taking 2 extra steps than optimal tour (0.6).

Case III - *step_pen* significantly lesser than *d_pen*. Also, *p_err* is much bigger (such that [1 - *p_err*] < *p_err*/3).

step_pen = -0.01, *d_pen* = -0.5, *p_err* = 0.9

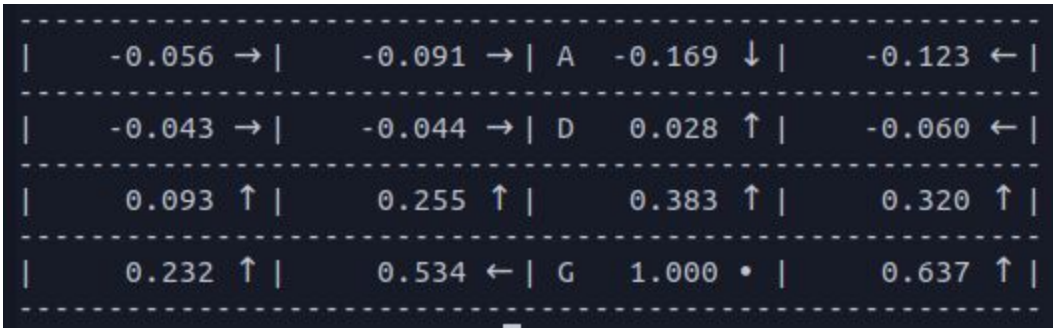


Fig e. Policy Iteration

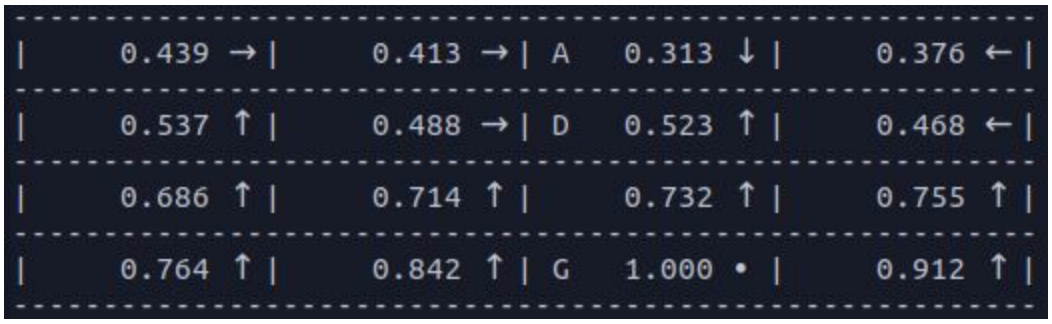


Fig f. Value Iteration

As it can be seen from the image, the policy tries to take anything other than the optimal policy obtained in Case I. This is because the sent action has only (0.1) probability to be received correctly, whereas the other 3 actions have a probability of (0.3) each. Hence, the sent action has to be anything but the optimal one!

VI. POLICY ITERATION VS. VALUE ITERATION

Environment Dimensions	Number of Iterations to converge	
	Value Iteration (VI)	Policy Iteration (PI)
3x3	7	5
4x4	7	6
10x10	13	11
25x25	29	31
50x50	55	61

From the table, it is clear that PI does better for smaller environments whereas VI takes over in bigger ones. This is due to the fact that in PI, the Policy is initialised randomly. Hence, there is some chance for some part of it to be optimal. The Policy is then evaluated and then re-assessed from the evaluation. Every iteration, the partly optimal policy induces more optimality in the policy. This is because V values are almost correctly computed from the parts of Policy which are optimal and these V values are again used to calculate the next Policy. Essentially, the parts which were optimal when initialised propagate to the rest and the whole Policy becomes optimal.

In VI, however, all values are initialised to 0, as we do not exactly know the possible expected rewards in each tile. They are gradually computed from the V values of the previous iteration.

Changes in PI with every iteration are discrete but they are continuous for VI. Therefore, in bigger environments, the headstart PI gets is neutralised by the more number of computations it has to do because it cannot propagate continuous values.

Hence, VI performs better in bigger environments and PI in smaller ones.

VII. CONCLUSION

The experiments clearly show that varying *step_pen* gives rise to different policies depending on whether stepping on detector plates is better than taking a few extra steps to avoid the detector plate. Also, fixing *p_err* to a high value gives an interesting result. The policy obtained is totally different than that obtained previously, as error has a higher chance than the correct signal to propagate.

In VI and PI, the latter does better in case of smaller environments and the former, in the case of bigger environments. This is due to the continuous scale of change in VI and also the partly optimal PI gets due to random initialisation.

Hence, VI should be used in case of bigger environments, which is generally the case.