COL 864: Planning and Estimation for Autonomous Systems

Assignment 2

Report submitted by

Ankit Garg **2017EE10439**

Karan Mittal **2017ME20671**

INDIAN INSTITUTE OF TECHNOLOGY DELHI

5th May 2021

Contents

Chapter 1

Solving an MDP

The grid world is given to us, as shown in the Figure 1.1. The shaded region corresponds to the walls present in the grid. The agent can move to any of its adjacent grid cell. We formulate it as an MDP problem as follows :-

- **Goal:** The red box (48,12) denotes the goal state.
- **Actions:** The agent can go North, South, East, West
- **States:** The 50x25 grid is the state space.
- **Transition Function:** 0.8 for the intended direction, 0.2/3 for rest of the directions
- **Reward Function:** (+100) is agent transitions into the goal. (-1) if it tries to go in the wall. (+0 otherwise).

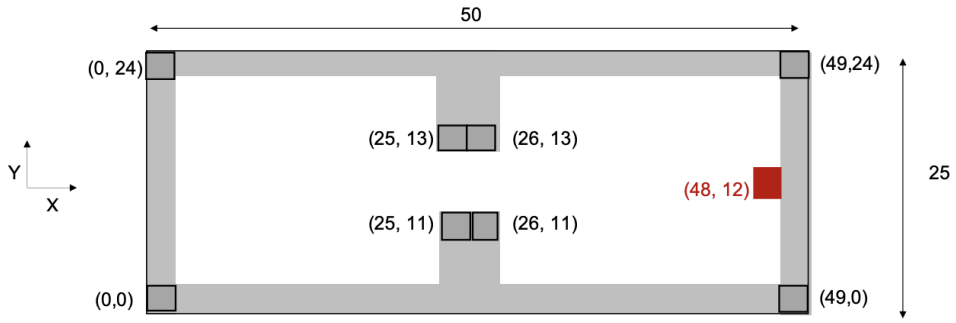

Figure 1.1: Grid World

1.1 Solve for a policy using value iteration

We wish to solve for the MDP formulated, in the previous section. We make use of the following equations

- **Bellman Equations:** The Bellman Equations are as shown in Figure 1.2. These equations relate the optimal state values. However these equations cannot be used directly to solve for optimal values.

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Figure 1.2: Bellman Equations

- **Value Iteration:** To find the state values, Bellman equations are converted into an iterative algorithm known as Value Iteration (Figure 1.3). This equation can be used to iteratively update values, until they converge. This is a dynamic programming solution. The convergence is said to be achieved, when the max-norm becomes less than a threshold value.

We use the Value Iteration algorithm. We simulated 100 iterations. The parameters used are :-

- Discount factor of $\gamma = 0.1$
- Threshold of $\theta = 0.1$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

Figure 1.3: Value Iteration

Value function obtained at the end of 100 iterations, plotted as a grayscale image can be seen in Figure 1.4. The value function is very high in value, near goal state. However due to high discount factor ($\gamma = 0.1$), the state value significantly drops as we move away from the goal.

We implement policy extraction, to obtain policy from the calculated value function previously. We can see the obtained policy in Figure 1.5

Figure 1.4: Value function $\gamma = 0.1$

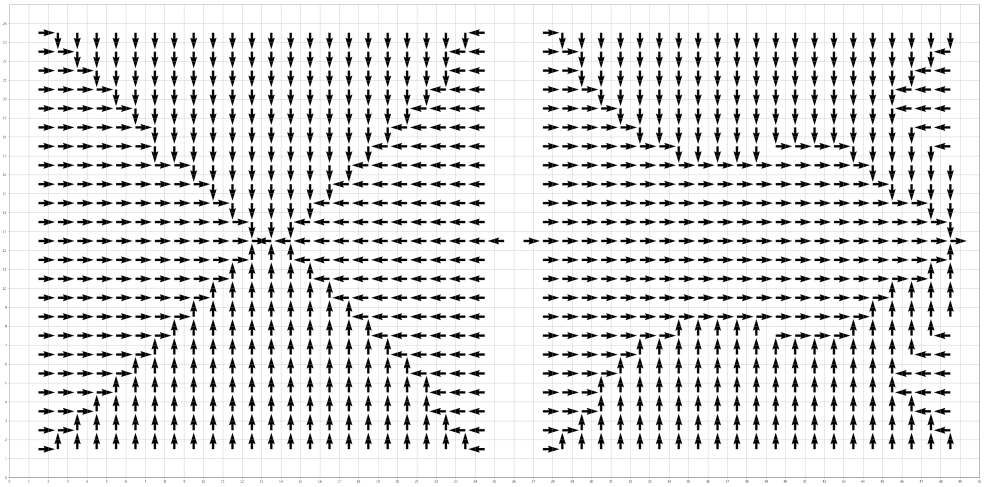

Figure 1.5: Policy $\gamma = 0.1$

1.2 Variation in discount factor (γ)

Now, we take the discount factor as 0.99, and study the evolution of value function at different time steps. The results can be seen in the Figure 1.9. We can observe that, as the iterations increase, the states farther away from the goal, start having higher values.

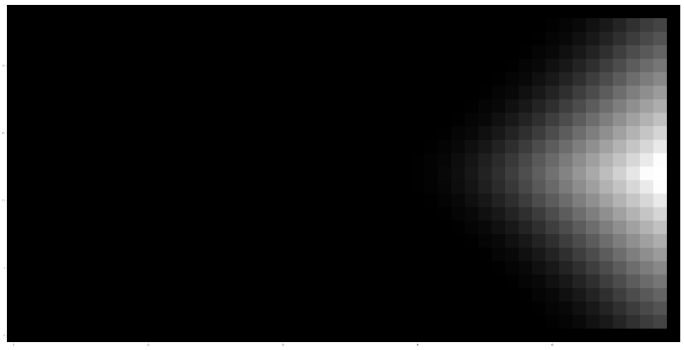

Figure 1.6: Iteration 20

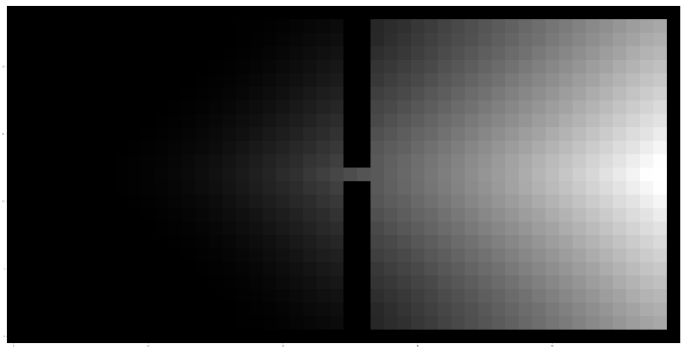

Figure 1.7: Iteration 50

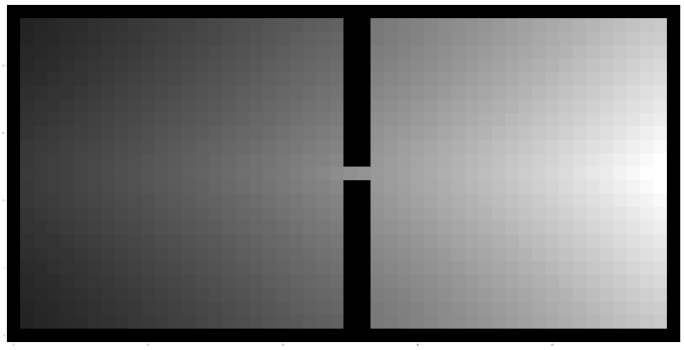

Figure 1.8: Iteration 100

Figure 1.9: Value function for $\gamma = 0.99$

The policy obtained after 100 iterations is shown in the Figure 1.10. Clearly, all the states in the state space have a goal directed, policy. Unlike the case of $gamma = 0.1$, in which the agent if present in the left half will not try to go towards the goal.

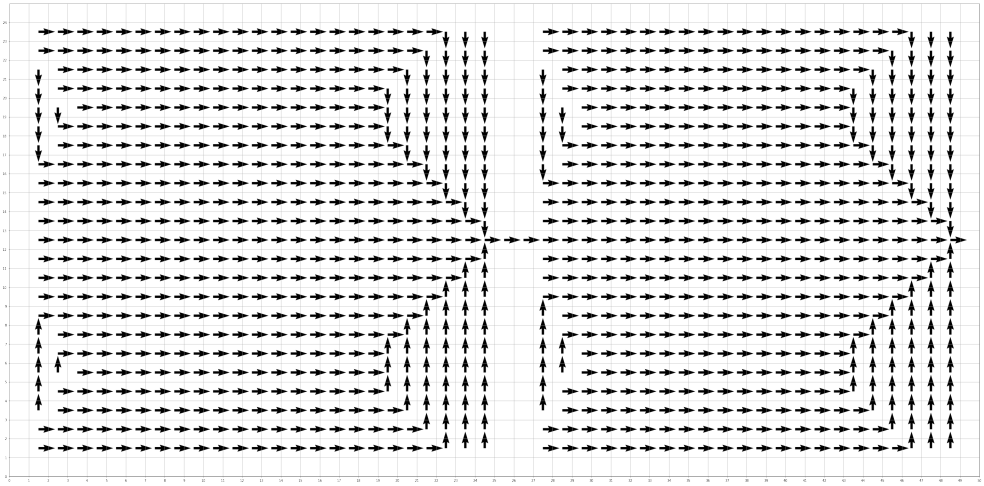

Figure 1.10: Policy for $\gamma = 0.99$ after 100 iterations

1.3 State Visitation Count

For $\gamma = 0.99$, we obtained the value function from our implementation of value iteration. From that we extracted policy. In this part, we executed this policy 200 times and calculated how many time a state is visited. The number of time-steps for each execution were fixed at 1000.

The start position of the agent is at (1, 1). The plots can be seen in 1.14. The Figure 1.11, clearly shows the path taken by the robot, under our calculated policy, when averaged over 200 policy executions. Once the agent reaches the goal state, it remains, there. And hence in Figure 1.12 and in Figure 1.13, the visitation count of goal state is very high.

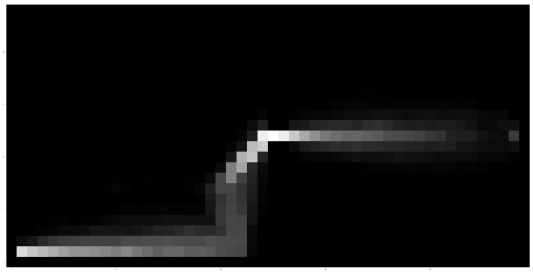

Figure 1.11: After 70 steps

Figure 1.12: After 100 steps

Figure 1.13: After 1000 steps

Figure 1.14: State Visitation count (obtained by executing policy 200 times)

1.4 Convergence of Value Iteration

We study convergence for different values of γ. The plots can be seen in the Figure 2.14. We observe that in case of $\gamma = 0.01$, the values converge much faster. This is because of the high discount factor, the value change in policy decreases less than threshold (0.1). However in the case of $\gamma = 0.99$, the value function does not converge in 100 iterations. This is because of low discount factor, the states, far away from goal, are still have updates greater than threshold.

We can also observe that, policy converges much faster, than the values. As you can see in the Figure 2.14. This is consistent with the empirical evidence, that policy converges faster than the values.

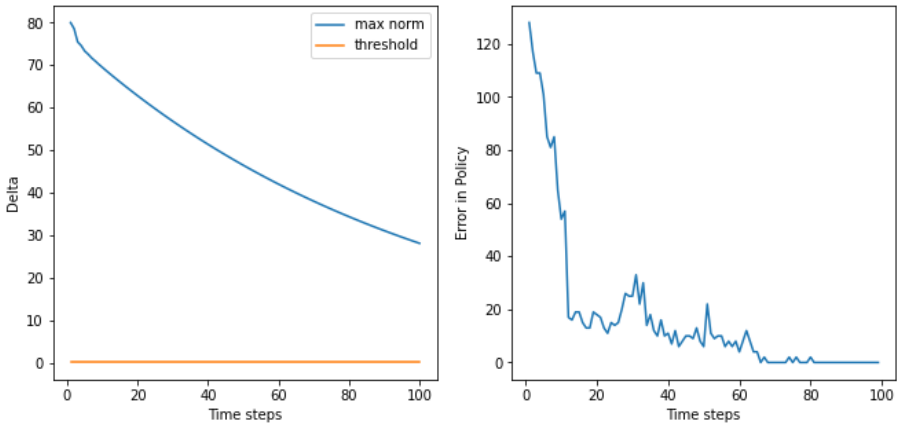

Figure 1.15: $\gamma = 0.99$

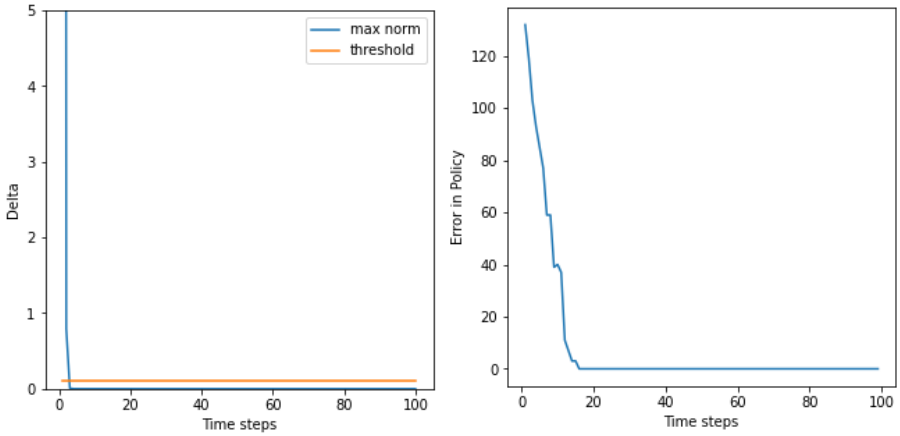

Figure 1.16: $\gamma = 0.01$

Figure 1.17: Convergence in values and policy for different γ

Chapter 2

Q-Learning

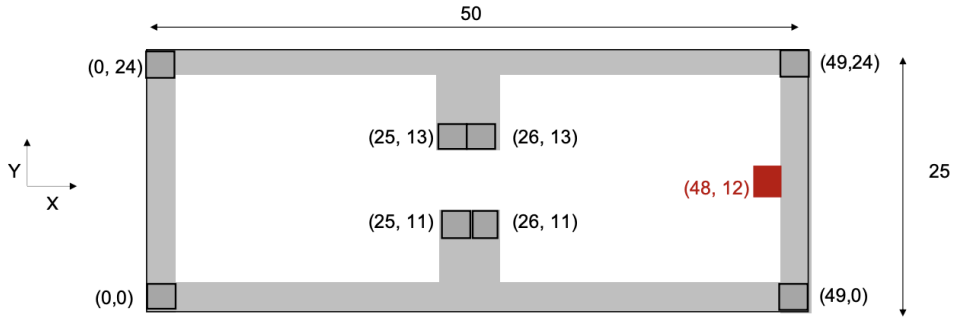

Figure 2.1: Grid World

The problem is same as the previous part. However this is posed as an RL problem. The reward function and transition functions are unknown to the agent. The agent experiences, various episode, and the idea is to able to learn a policy from this episodes. There are various algorithms that can be used. We implement Q-learning algorithm.

2.1 Implement Q-Learning

This is one of the core algorithms of reinforcement learning. After experiencing a tuple (s,a,r,s,a). It used the update rule as specified in the Figure 2.2.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[(r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t) \right].$$

Figure 2.2: Q Learning update equation

During an episode the action is chosen using the epsilon-greedy exploration algorithm. It is given in the Figure 2.3

And finally when the Q function is learned, we extract the policy using the equation shown in the Figure 2.4

We use exploration parameter $epsilon = 0.05$ and have learning rate $\alpha = 0.9$ initially. The learning rate decays by 0.08, after every 1250 episodes. We take 10000 episodes, with each episode having 1000 time steps. The start states for each episodes are picks at random.

$$\pi_\epsilon(s) = \begin{cases} \max_a Q(s,a) & \text{with probability}(1-\epsilon) \\ \text{RAND}(A) & \text{otherwise} \end{cases}$$

Figure 2.3: ϵ-greedy exploration

$$\pi(s) = \max_a Q(s,a)$$

Figure 2.4: Policy Extraction

2.2 Visualise state value function & policy

We learn the Q function using the equations described in the previous section. We convert state action values to state value function, with the help of Bellman Equation (described in previous part). The state function obtained is plotted as in Figure 2.5

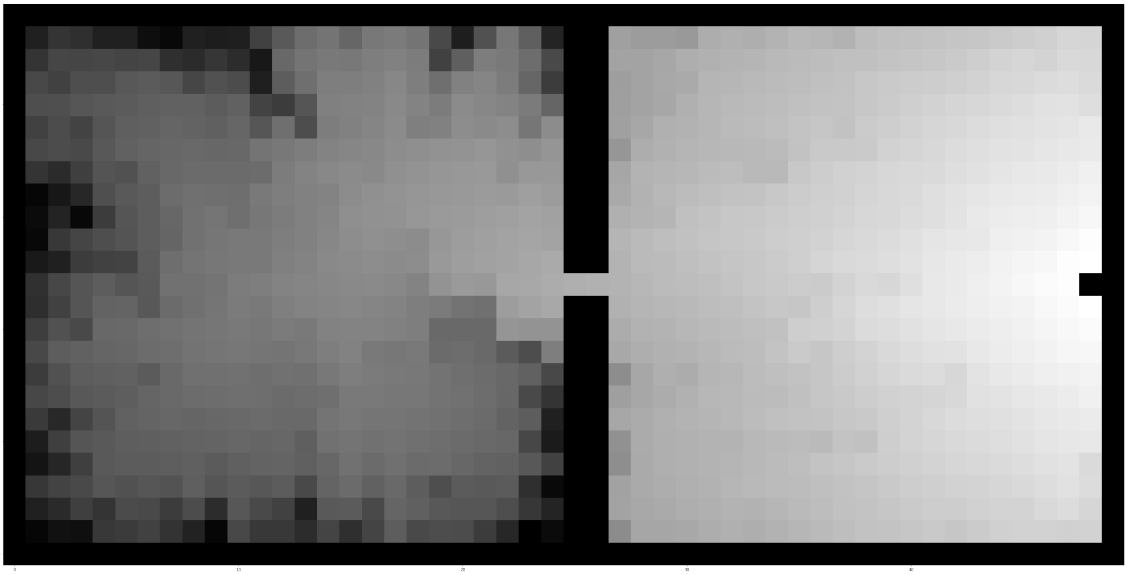

Figure 2.5: Value function ($\epsilon = 0.05$)

The policy is extracted using the Equation shown in the Figure 2.4 The policy can be in the Figure 2.6

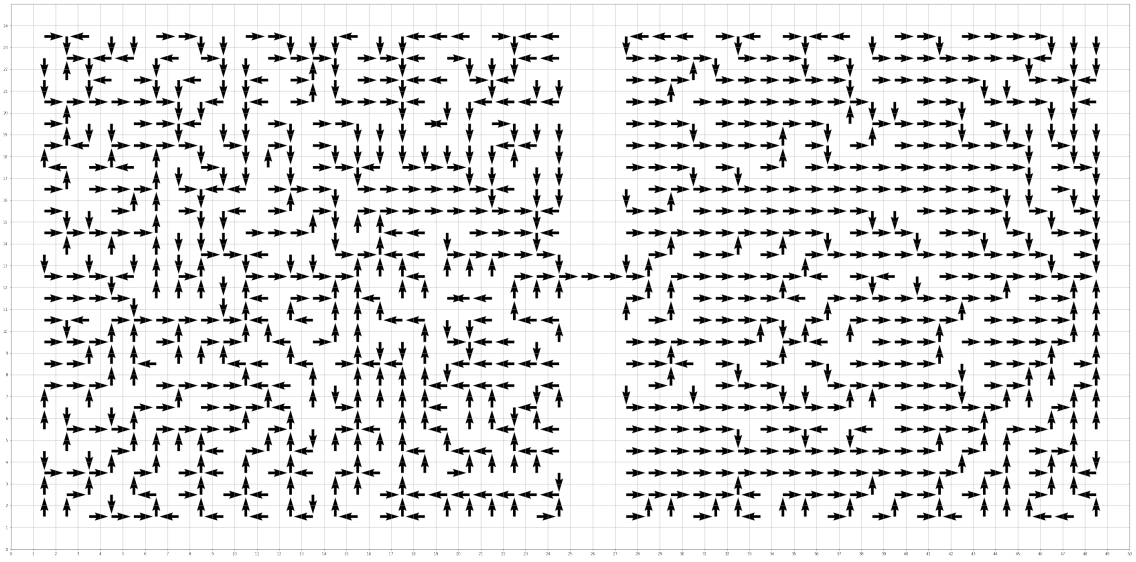

Figure 2.6: Policy ($\epsilon = 0.05$)

2.3 Effect of exploration parameter (ϵ) on Q-Learner

The value ϵ decides the trade-off between exploration and exploitation. Higher value of ϵ implies higher exploration as compared to exploitation. As we can see in case of $\epsilon = 0.5$ from Figure 2.7 and Figure 2.11, the agent will be able to reach the goal from any cell in the grid. In case of a lower $\epsilon = 0.05$ (Figure 2.8 and Figure 2.12), the agent will be able to reach the goal from any cell in the right half and from some cells in the left half of the grid. When the exploration parameter was set to as low as $\epsilon = 0.005$ (Figure 2.9 and Figure 2.13), the agent is unable to explore the right half from the left half of the grid, implying that the agent will be able to reach the goal only if it is present in the right half of the grid.

The logical explanation for the above results is that when the value of ϵ is low, the agent does not explore enough. Therefore, when it is in the left half it starts exploiting the incentive of going away from the wall rather than exploring and reaching to the goal. When the value of ϵ is high, the agent manages to explore the path towards the goal instead of just exploiting the already known less optimal reward.

2.4 Reward Accumulated per Episode

We observe the reward accumulated per episode for $\epsilon = 0.5$ and $\epsilon = 0.05$ in Figure 2.15. The reward per episode converges earlier for $\epsilon = 0.5$, as it explores more in the grid and ends up finding the optimal path before $\epsilon = 0.05$. When both of them converge, the reward per episode is always higher for $\epsilon = 0.05$ because it exploits the already given optimal policy and in case of $\epsilon = 0.5$, it still continues to explore even after achieving the optimal policy.

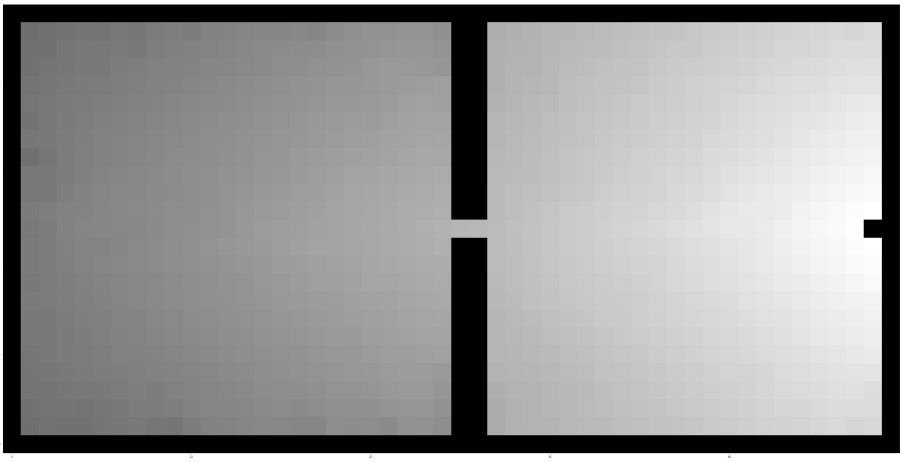

Figure 2.7: $\epsilon = 0.5$

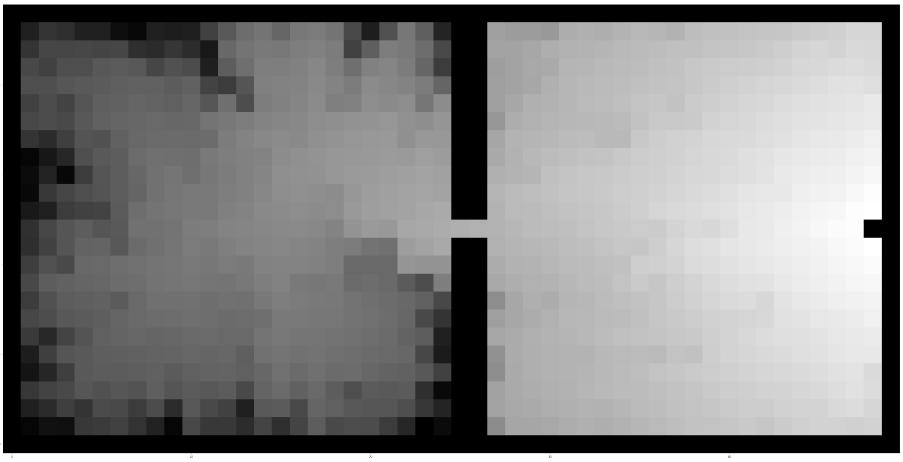

Figure 2.8: $\epsilon = 0.05$

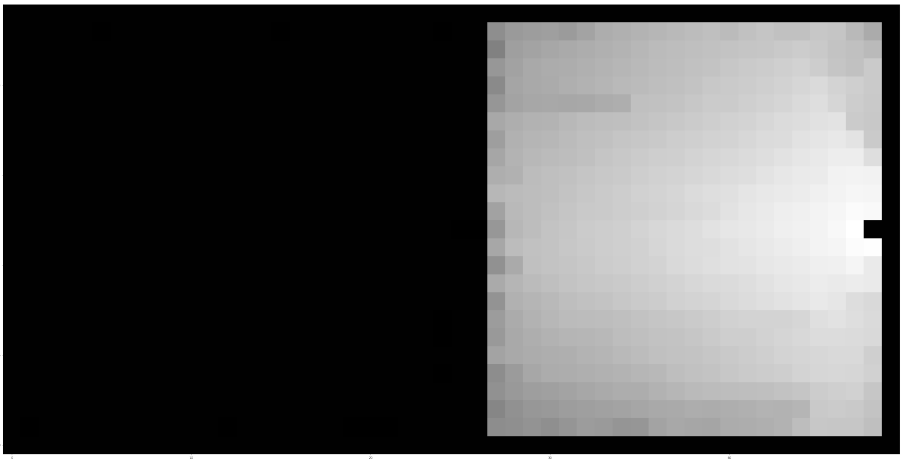

Figure 2.9: $\epsilon = 0.005$

Figure 2.10: Value function for different ϵ

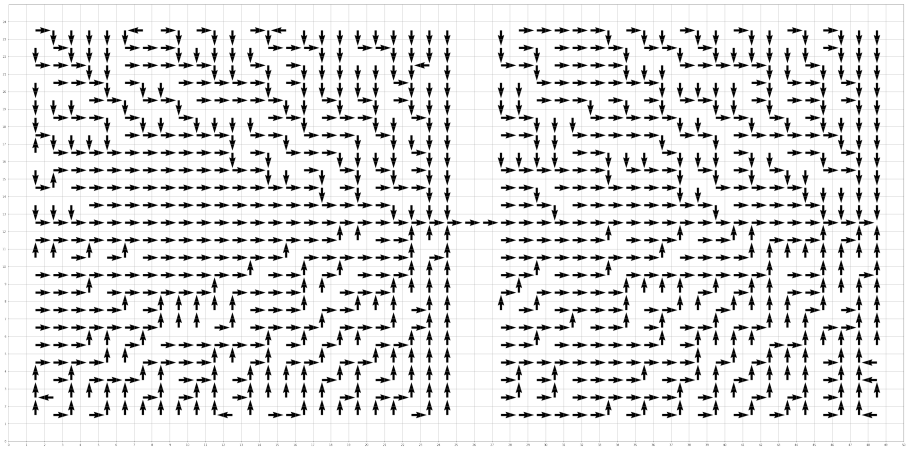

Figure 2.11: Policy ($\epsilon = 0.5$)

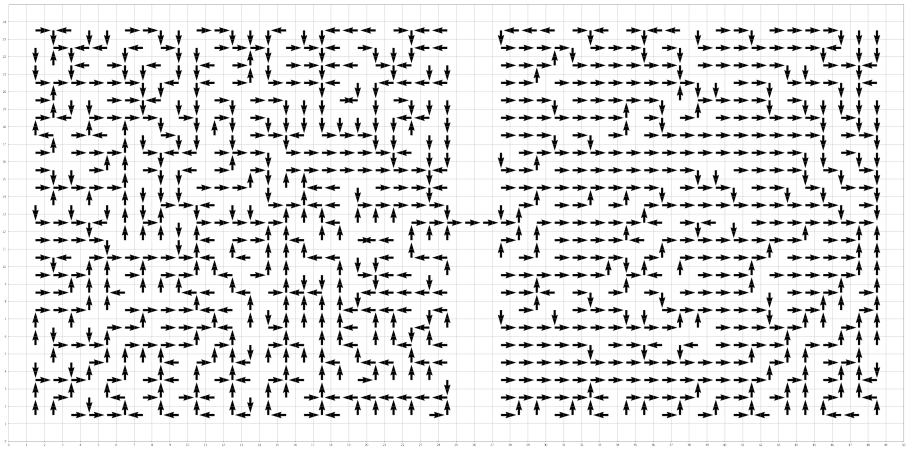

Figure 2.12: Policy ($\epsilon = 0.05$)

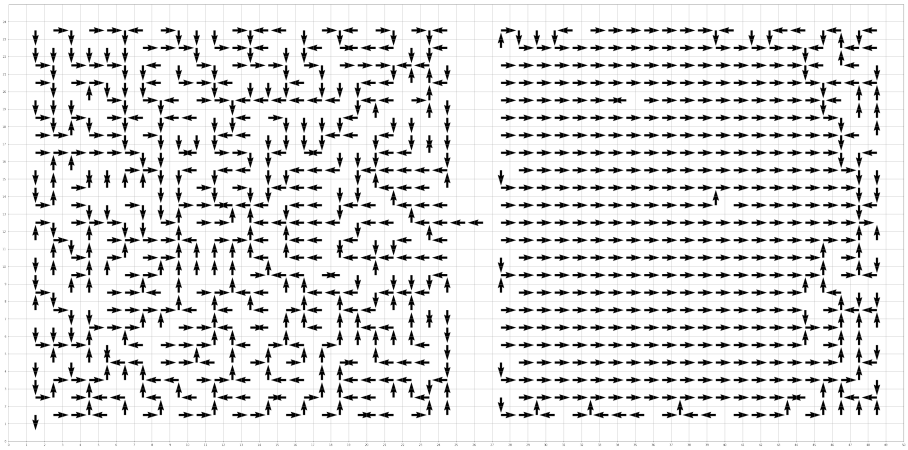

Figure 2.13: Policy ($\epsilon = 0.005$)

Figure 2.14: Policy for different ϵ

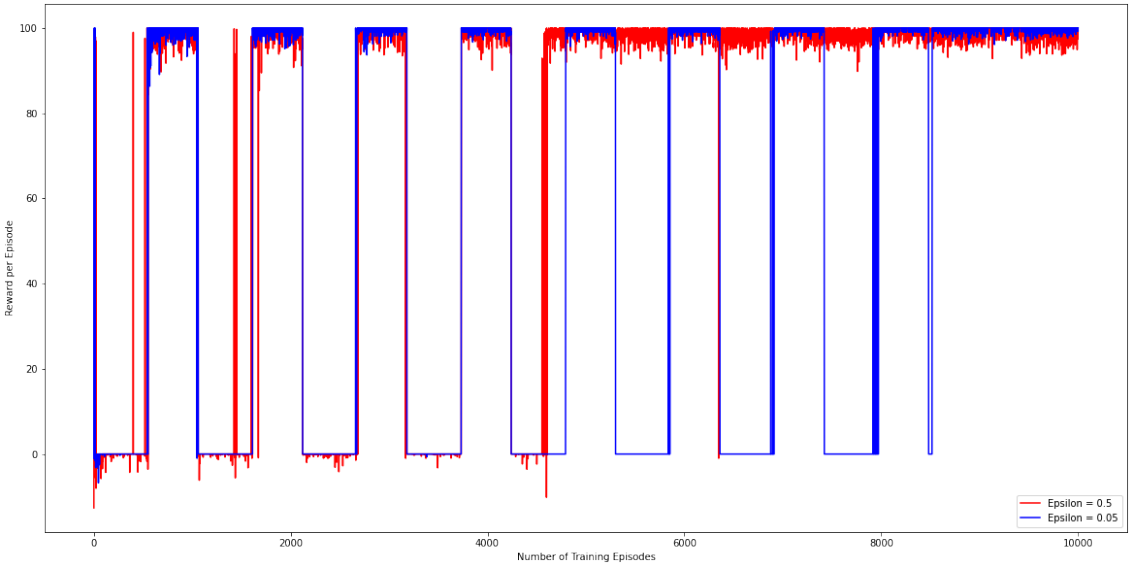

Figure 2.15: Reward Accumulated per Episode

Chapter 3

Conclusion

3.1 Solving an MDP

We implemented the Value Iteration Algorithm (with the help of Bellman Equations) to solve the given MDP and extract the optimal policy. We varied the discount factor γ and analysed the corresponding value function and policy. We also studied the convergence of value iteration for γ=0.01 and γ=0.99. The results obtained were quite intuitive, and have a logical explanation. The individual conclusion to each of the sub-parts has been included in the Chapter 1.

3.2 Q-Learning

We implemented Q-Learning algorithm on the problem given in Chapter 1. The only difference is that this time the robot does not have access to the transition function and the reward model. We varied the exploration parameter ϵ and analysed the corresponding value function and policy. We also studied the reward accumulated per episode for $\epsilon = 0.5$ and $\epsilon = 0.05$. The results obtained were quite intuitive, and have a logical explanation. The individual conclusion to each of the sub-parts has been included in the Chapter 2.

Bibliography

[1] Stuart J. Russell and Peter Norvig. 2003. Artificial Intelligence: A Modern Approach (3rd ed.). Pearson Education.

[2] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, Cambridge, MA, USA, 1st edition, 2005.