Reinforcement Learning Report

Question 1:

1. State which method you choose to solve the problem and justify your choice

The Value Iteration algorithm is used here to implement the DP agent because it integrates policy improvement and policy evaluation into a single step. This approach is more direct and can be faster. Value Iteration is also guaranteed to converge to the optimal policy and value function. Given the 500-step limitation, Value Iteration can converge more quickly within a constrained timeframe.

The threshold for terminating the iteration at 0.0001, which is sufficiently small to ensure precise convergence of the value function.

2. Graphical representation (γ = 0.88 and p = 0.9)

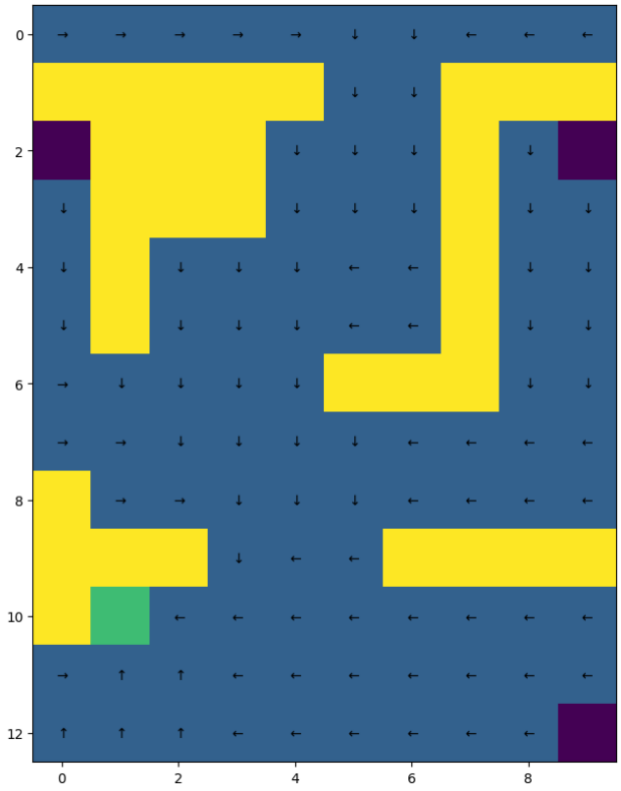

Figure 1 Graph of Optimal Policy Using DP Agent

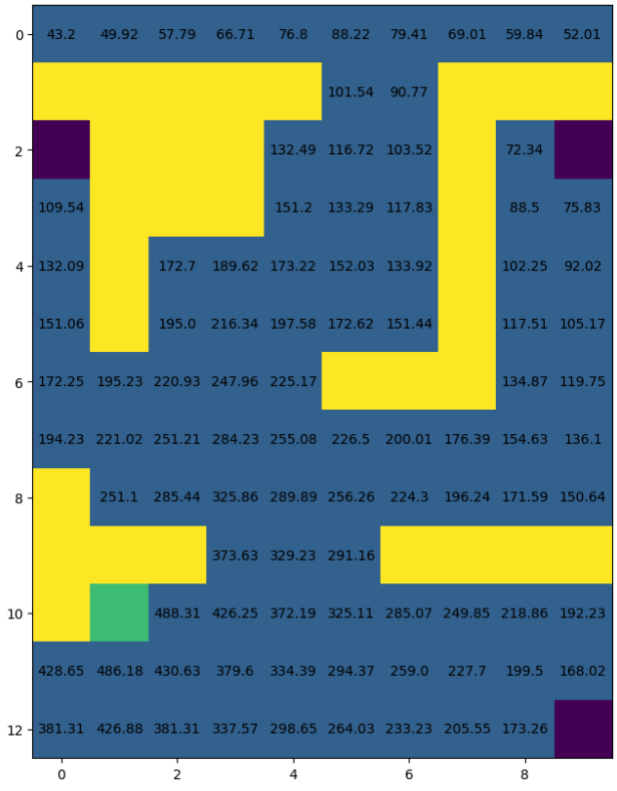

Figure 2 Graph of Optimal Value Function using DP Agent

3. Discuss the value of your γ and p

When p is too low (such as 0.1), the agent tends to move in the opposite direction because the probability of failure is higher than that of success. At a probability of 0.25, there is an equal chance of moving to any direction, leading the agent to behave completely randomly since all directions have the same probability of success. Beyond 0.25, we observe that the agent begins to converge towards an optimal policy. The values near the goal become higher than those at the edges, and the values around the absorbing states are very small or even negative. The current probability p in the code is approximately 0.88, as calculated using the CID number (refer to Figures 1 and 2), and this works better than when p = 0.4. When p is larger, the agent requires fewer iteration step to converge (Figure 8), because it saves the computation wasted on incorrect movement.

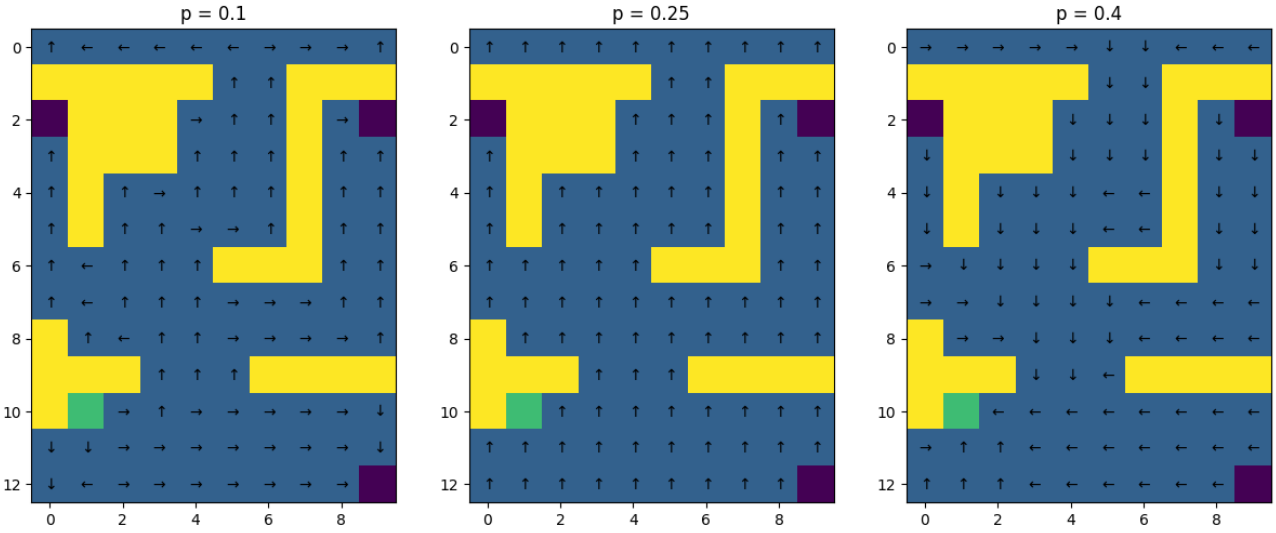

Figure 3 Graph of Optimal Policy Under P = 0.1, 0.25 and 0.4

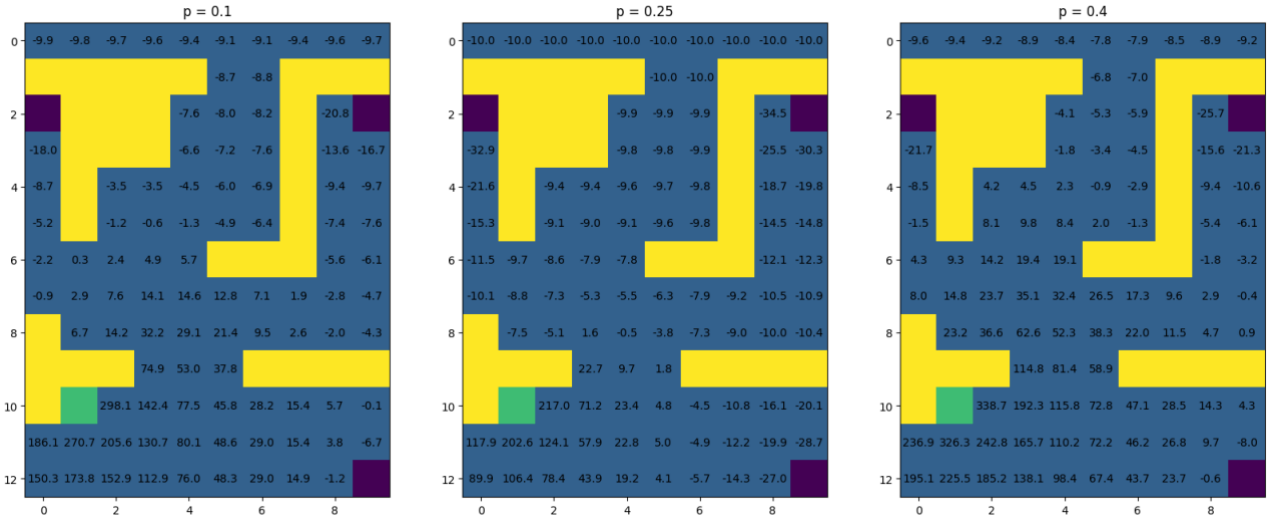

Figure 4 Graph of Optimal Value Function under P = 0.1, 0.25 and 0.4

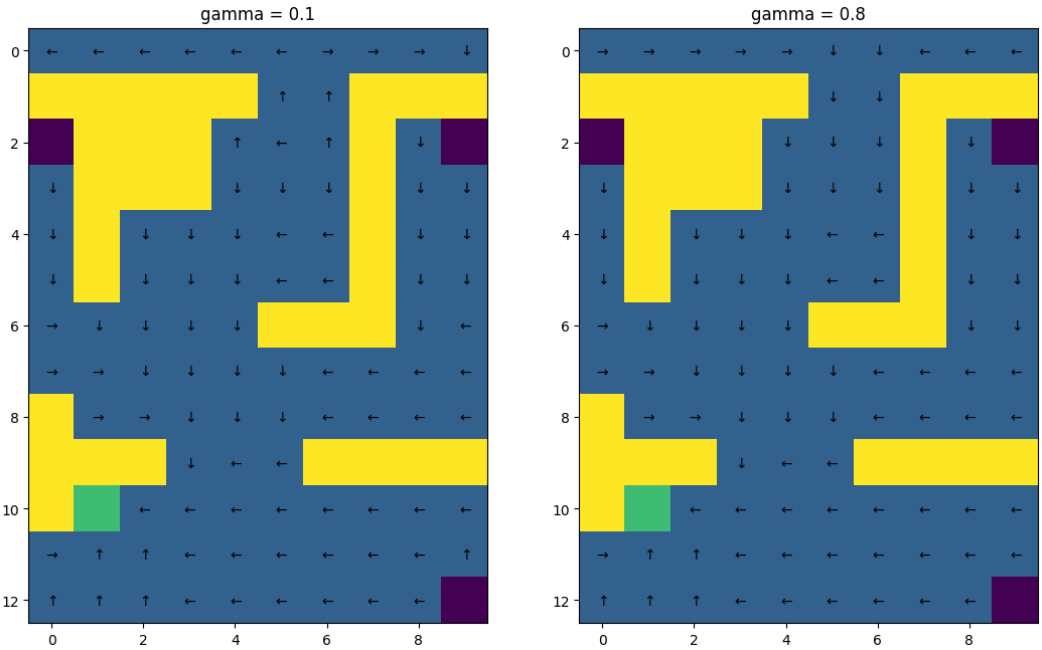

Figure 5 Optimal Policy Under γ = 0.1 and 0.8

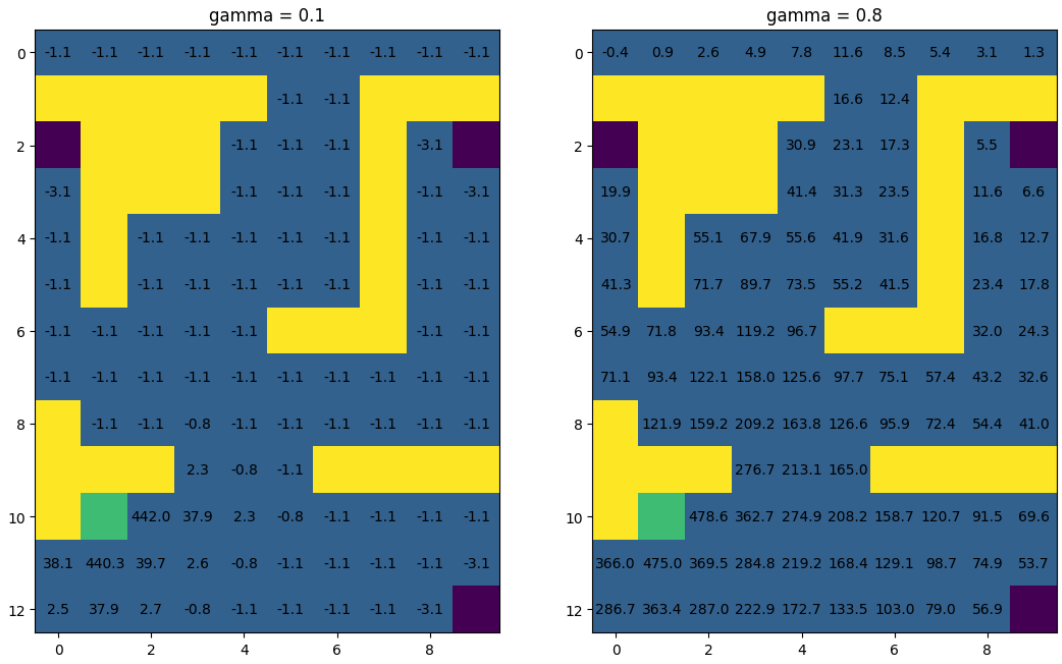

Figure 6 Optimal Function Under γ = 0.1 and 0.8

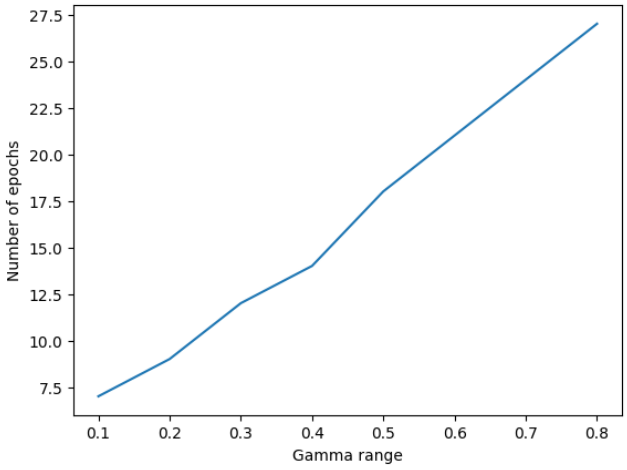

Figure 7 Number of Iteration until Convergence for Different γ

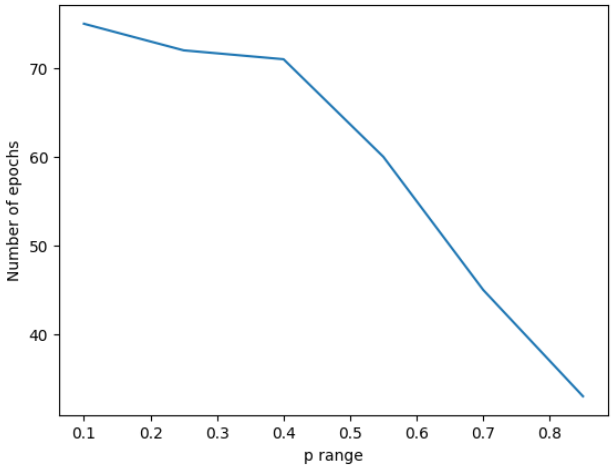

Figure 8 Number of Iteration until Convergence for Different p

From Figure 5 and 6, when γ is very low, the agent fails to generate an optimal policy, even though the distribution of its value function tends to be higher around the goal. Additionally, as illustrated in Figure 7, when γ is low, the agent converges more quickly. This is because a low gamma indicates a preference for immediate benefits over long-term rewards, which leads to suboptimal policy. Conversely, when the discount factor γ is set to 0.8, the agent may take longer to converge, but it values future rewards nearly as much as immediate ones. This can be beneficial in environments where planning over many steps is important, such as in maze.

Question 2

1. State which method you choose to solve the problem and justify your choice.

The On-policy ϵ-greedy First-visit Monte Carlo Control Algorithm is employed because of its efficacy in exploration and learning from unknown environments. This algorithm usually selects the best-known action, while an exploration rate (ϵ) of 0.3 allows for a random action selection. This method facilitates exploration and avoid convergence to local optima. A total

of 4000 episodes ensures that the agent has sufficient traces to converge to an accurate value function. The first-visit approach enhances the learning efficiency by only considering the first state-action occurrence in each episode.

2. Graphical representation (ϵ = 0.3, γ = 0.88 and number of episodes = 4000)

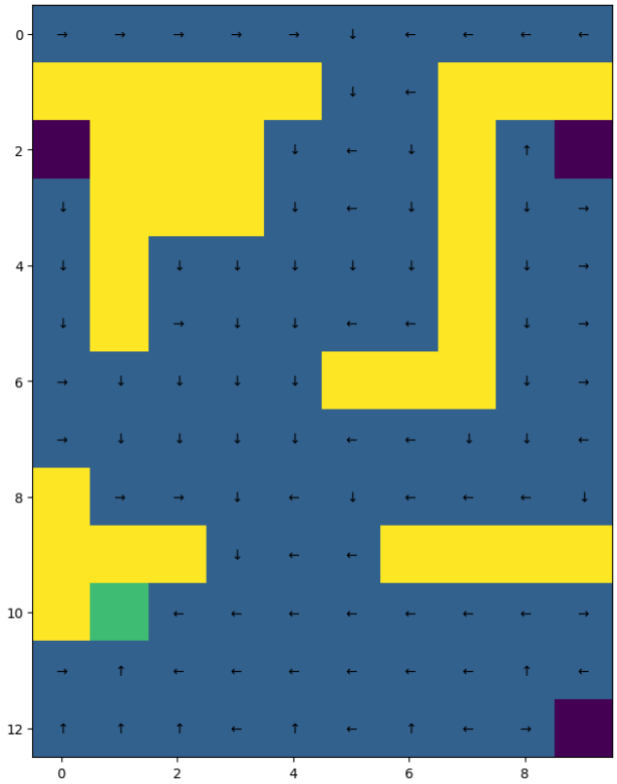

Figure 9 Optimal Policy from MC Agent

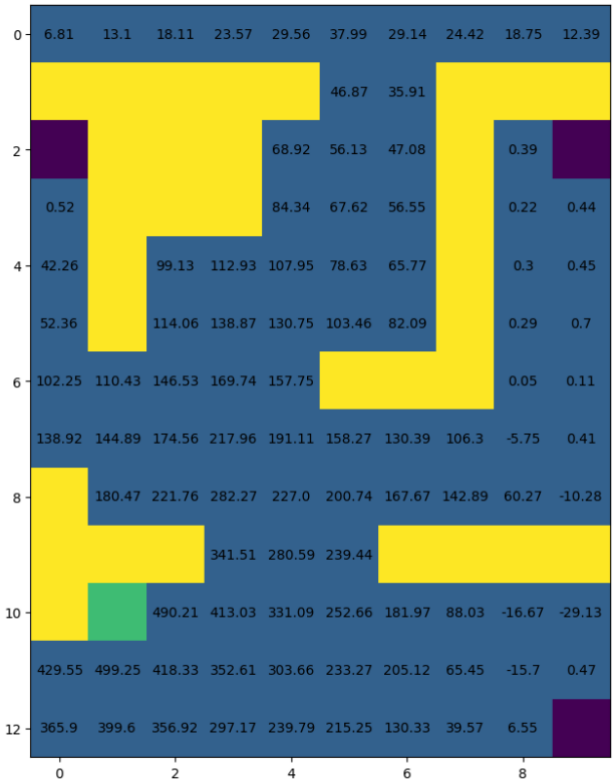

Figure 10 Optimal Value Function from MC Agent

3. Plot the learning curve of your agent (after training 25 runs)

We can observe that after 2000 episodes, the agent begins to converge, as indicated by the stabilization of the mean rewards and the reduction in standard deviation. The MC agent requires more time to converge than the TD agent, which may be due to the limitations of the on-policy approach. It tends to follow previously visited states with higher values, leading to suboptimal exploration. While the epsilon parameter does contribute to exploration, it is not sufficient to fully address this issue.

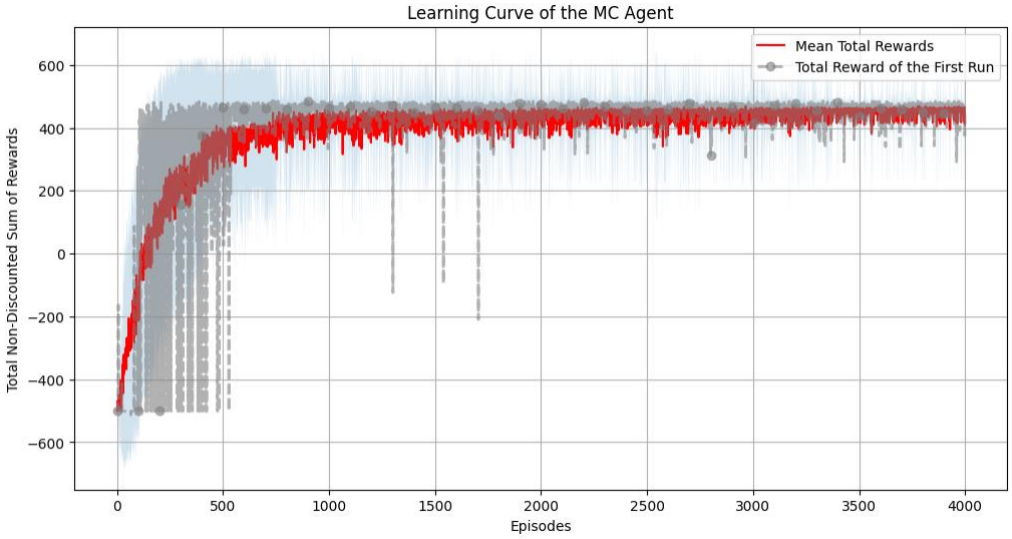

Figure 11 Learning Curve of the MC Agent with the Mean and the Standard Deviation of the Total Rewards

Question 3:

1. State which method you choose to solve the problem and justify your choice

The Q-learning algorithm is employed because it is more robust than on-policy methods. The agent learns from observing the actions taken by another policy during exploration. Compared to on-policy methods, off-policy methods are more likely to explore a wider range of the environment and can learn from any action taken, not just those taken by the current policy. This ensures a wide exploration of the state space while still acquiring knowledge about the optimal policy, which can lead to faster convergence. The policy is updated after each episode, which is a more computationally efficient approach.

The learning rate α is set to 0.3, which indicates how much new information overrides existing knowledge. This value keeps a balance between rapid learning and preventing the agent from overfitting to recent changes in the environment.

An epsilon value ϵ of 0.3 indicates a 30% chance of selecting a random action, as opposed to the best action suggested by the current Q-values. This facilitates the exploration of the state space.

The decision to run 1000 episodes assumes that this number is sufficient for the agent to learn accurate Q-values. The Q-learning algorithm can converge to the optimal policy within these episodes.

2. Graphical representation (ϵ = 0.3 and α = 0.3)

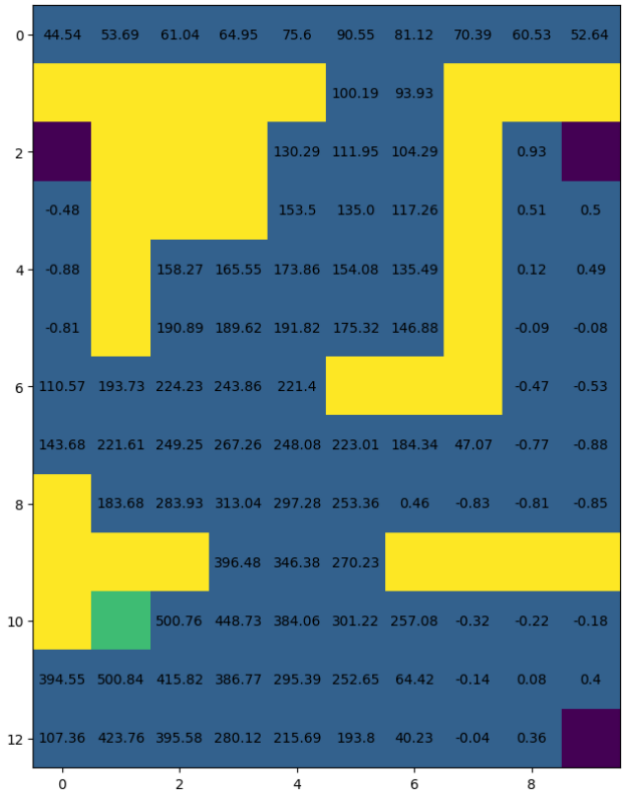

Figure 12 Optimal Value Function from TD Agent

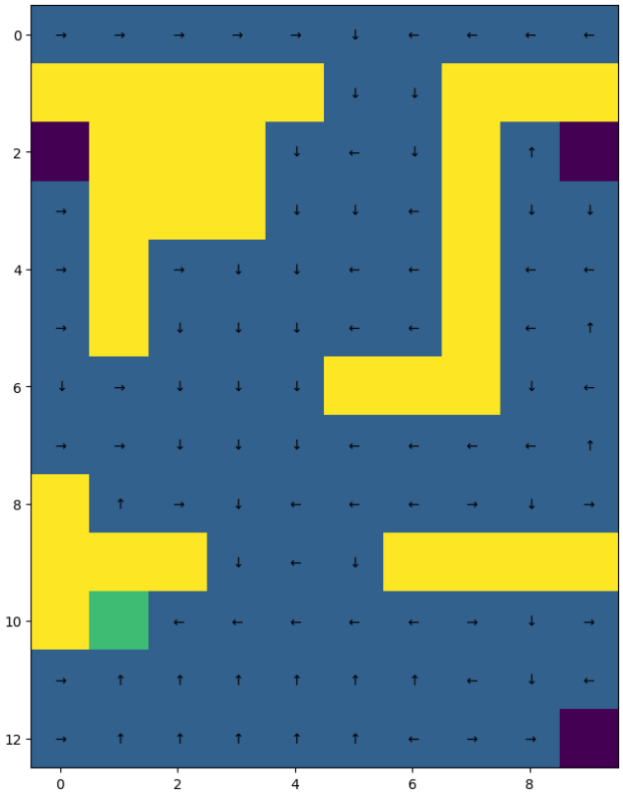

Figure 13 Optimal Policy from TD Agent

4. Discuss exploration parameter ϵ and the learning rate α of your algorithm

When the learning rate (alpha) is set to 0.1, the learning curves demonstrate that the agent converges faster with smaller epsilon (ϵ) values, such as 0.1 and 0.3, compared to when ϵ is set to 0.9. When ϵ is close to 1, the agent prioritizes exploration over exploitation, often selecting actions randomly. This approach is inefficient for long-term learning as the agent

may not effectively utilize its knowledge. However, epsilon should not be too small either. As it nears zero, the agent tends to exploit its knowledge and explores less. This means it will always choose the action that it currently believes to have the highest value, based on experience. This can lead to suboptimal policy if the agent hasn't explored enough to make an accurate Q value, which is not the case of the current implementation though.

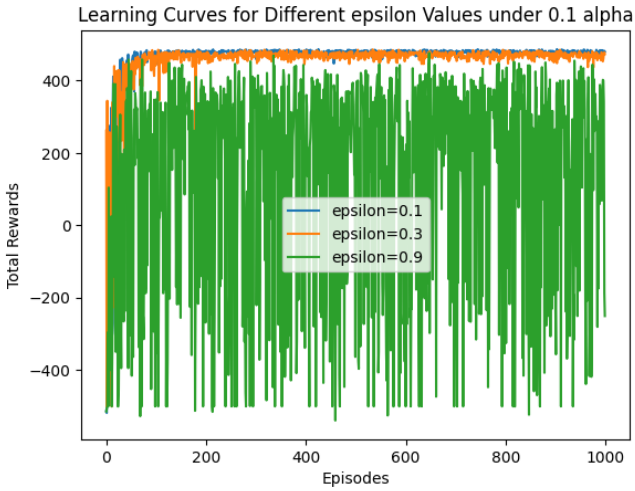

Figure 14 Learning Curve for Different ϵ when α = 0.1

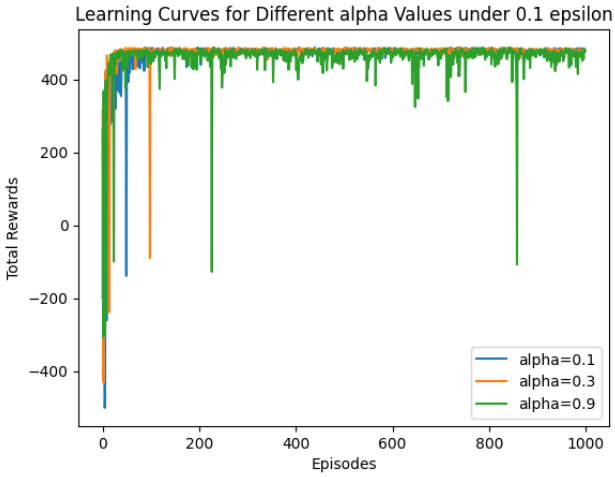

Figure 15 Learning Curve for Different α when ϵ = 0.1

When ϵ = 0.1, with different learning rate α, the learning curve shows different speed and stability of convergence, due to the varying impact of new information on the Q-value updates.

When the α is set to 0.1, the agent updates its Q-values slowly, relying heavily on current knowledge, which leads to stable learning and lower convergence to the optimal policy.

With a moderate α of 0.3, the updates are more balanced, allowing the agent to learn at a reasonable pace without heavily discounting previous information. The convergence is slightly steadier and quicker.

When α = 0.9, it results in rapid updates to the Q-values, prioritizing new reward and estimate significantly over the old ones. This approach can lead to less stable learning, as the Q-values may fluctuate significantly. This method can potentially lead to divergence or oscillation around the optimal policy, rather than convergence.

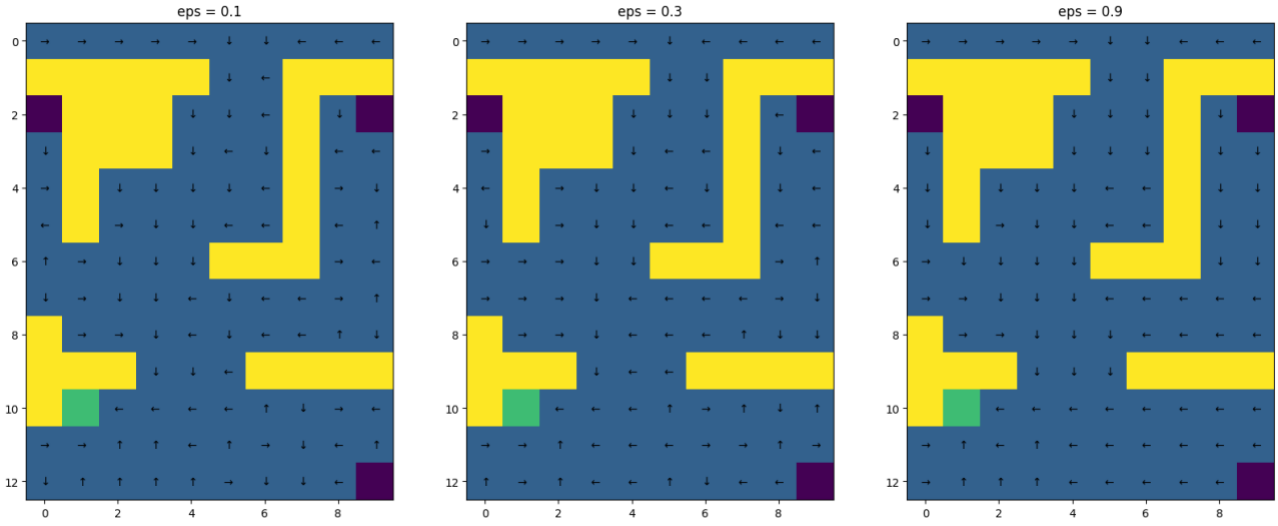

Figure 16 Optimal Policy for Different ϵ when α = 0.1

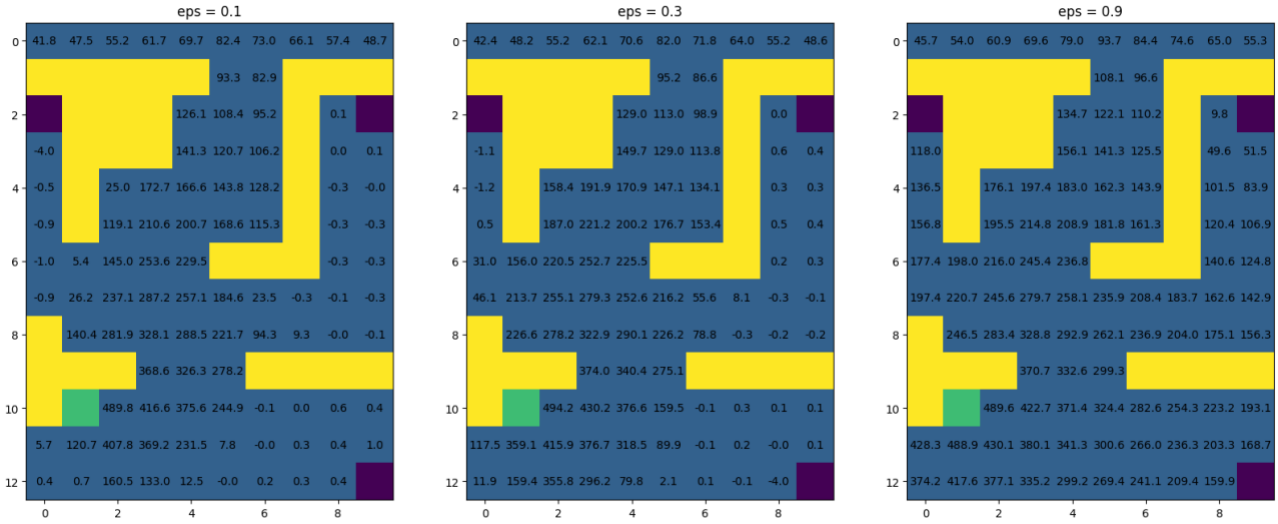

Figure 17 Optimal Value Function for Different ϵ when α = 0.1

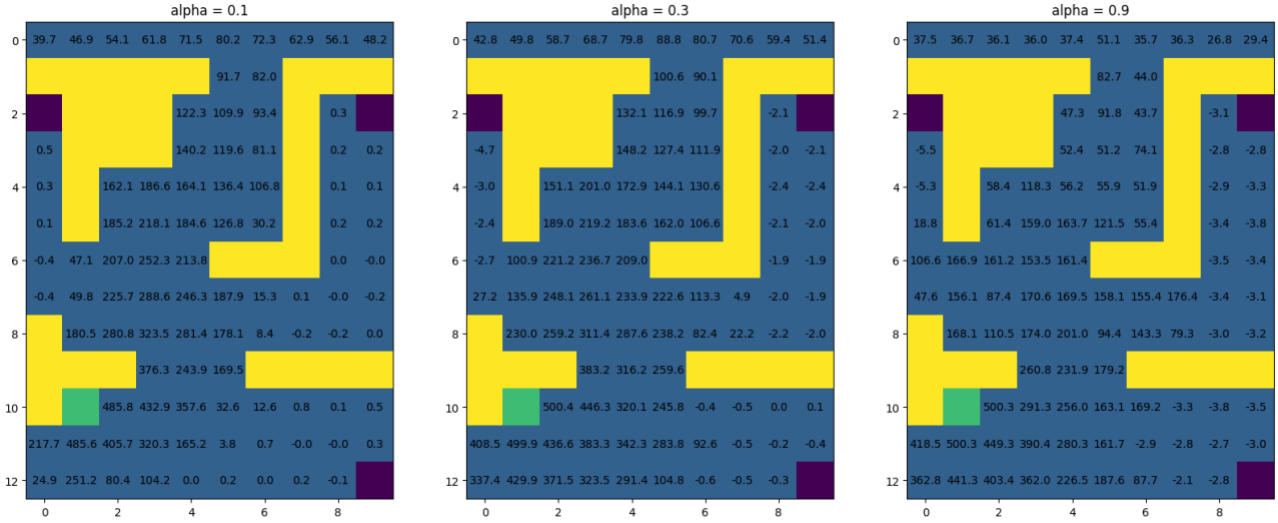

Figure 18 Optimal Value Function for Different α when ϵ = 0.1

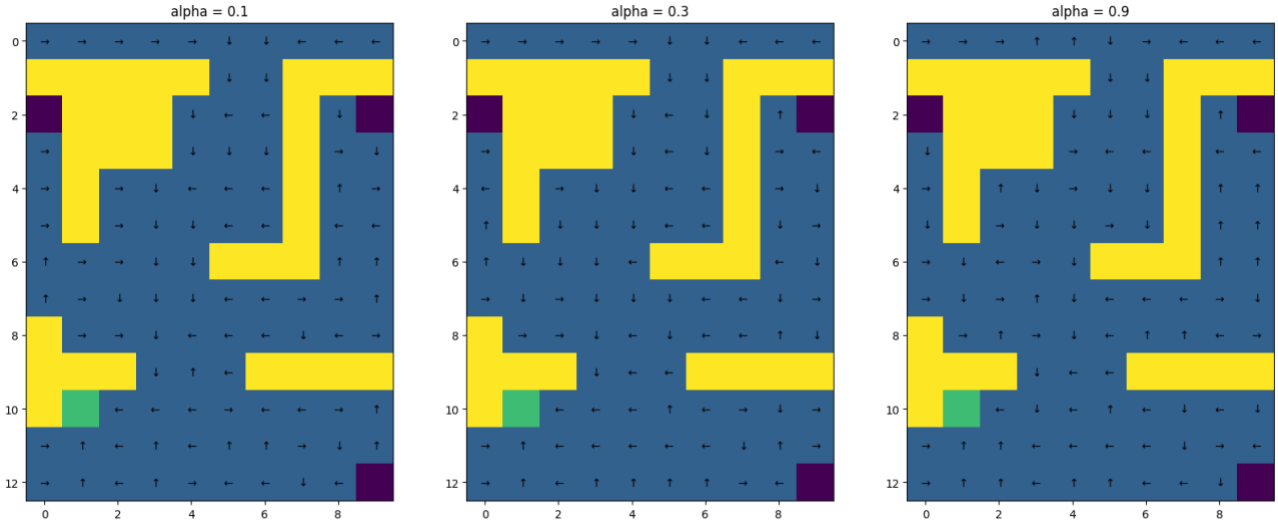

Figure 19 Optimal Policy for Different α when ϵ = 0.1

Despite the variations in α and ϵ, this experiment still results in optimal policies for all settings in the end (Figure 16 – 19). This outcome possibly because after 1000 episodes, each combination eventually provides sufficient exploration to visit all relevant state-action pairs, and enough exploitation to refine the policy to optimality.