Solving the Maze problem with Reinforcement Learning

1. Dynamic Programming

(a) To solve the problem, we use the value iteration method. We set our sole parameter, **threshold** = 0.001, as a stopping criterion. This criterion is applied when δ (calculated as the difference between the current and previous values of V) exceeds this threshold, as it has been empirically shown to ensure convergence.

(b)

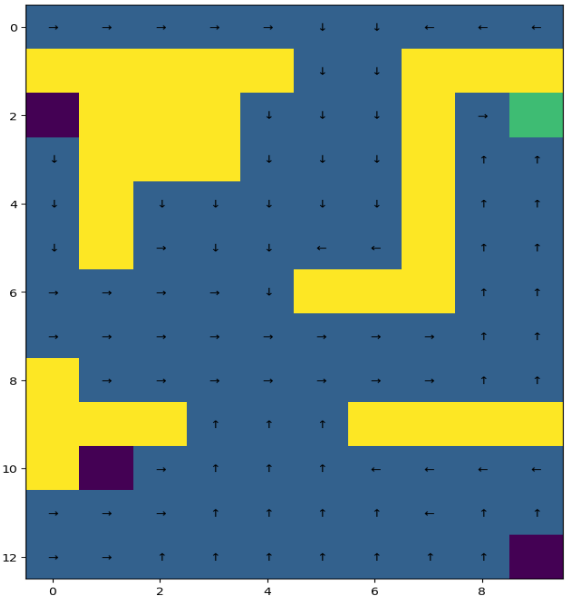

Figure 1: Optimal policy of Dynamic Programming.

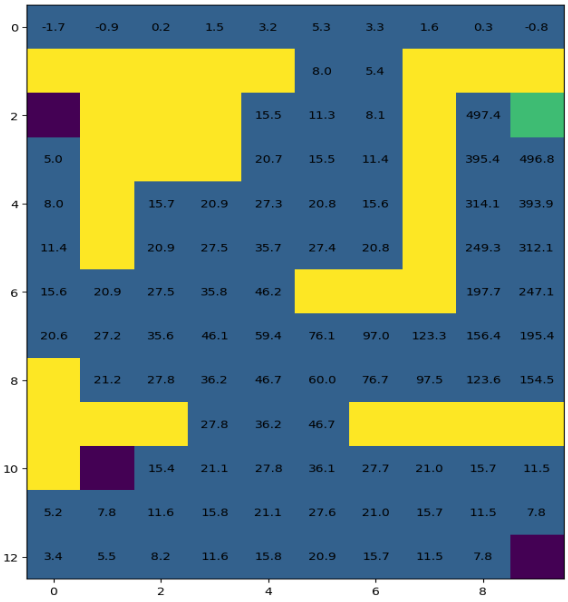

Figure 2: Optimal value of Dynamic Programming.

(c) What is the effect on the optimal policy and value function when rewards are scaled by a factor of 2?

If rewards are scaled by a factor of 2, the optimal policy would remain unchanged because all values of the state-action pairs would undergo the same transformation, preserving their relative values. Conversely, the optimal value function would be doubled, as all Q values would be multiplied by 2, and the optimal value function is derived from the maximum Q value.

(d) Suppose you now wish to design the reward function such that, for any state s, the value for that state is equal to the probability of eventually reaching the goal state, when starting in state s.

For this function, the discount factor has to be $\gamma = 1$ in order to represent real probabilities and not discounted ones. For all non-terminal states s and all non-goal terminal states, we would set the rewards as follows:

$$R(s, a) = R(\text{terminal}, a) = 0$$

since we have not reached the goal. On the other hand, for the goal state, we would have:

$$R(\text{goal}, a) = 1$$

since we have reached our goal. The resulting V function would be:

$$V(s) \leftarrow \begin{cases} 0 & \text{if } s \text{ is absorbing and not the goal} \\ 1 & \text{if } s \text{ is the goal} \\ \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s, a) + V(s') & \text{otherwise} \end{cases}$$

2. Monte-Carlo Reinforcement Learning

(a) To solve this problem, we have chosen the on-policy ϵ-greedy first-visit Monte Carlo control algorithm. In this approach, we needed to set two parameters: ϵ and the number of episodes. For ϵ, we observed empirically that setting $\epsilon = \frac{1}{k}$ led to very fast convergence, which limited the agent's ability to explore the environment. To allow for more gradual exploration, we chose to decrease ϵ more slowly by updating it as $\epsilon = \epsilon \times 0.99$. Additionally, to ensure that the agent always has some exploration probability, we applied the following condition:

$$\epsilon = \max(0.2, \epsilon \times 0.99)$$

This prevents the agent from getting stuck in certain states. For the number of episodes, we set it to 300, as the learning curve of the agent indicated that it ceased to improve beyond this point.

(b)

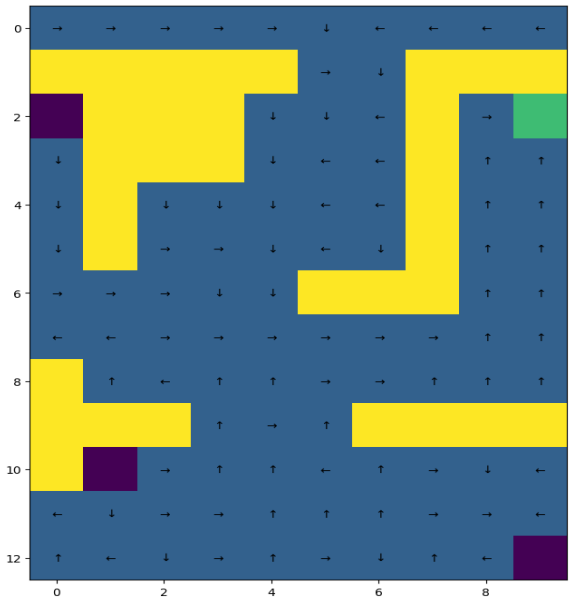

Figure 3: Optimal policy of Monte-Carlo.

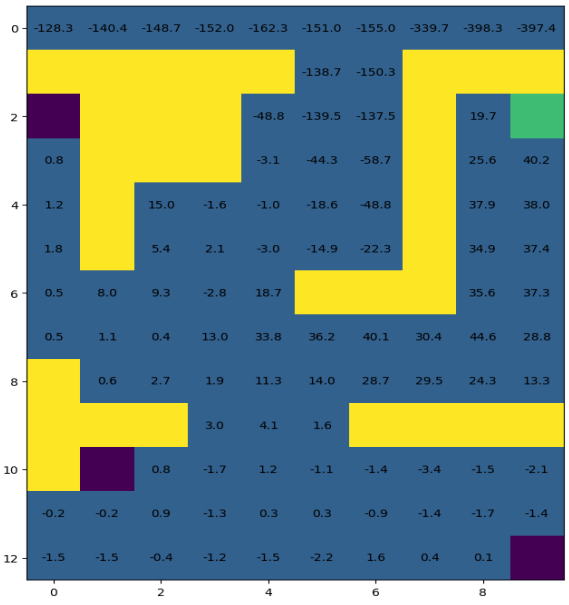

Figure 4: Optimal value of Monte-Carlo.

(c)

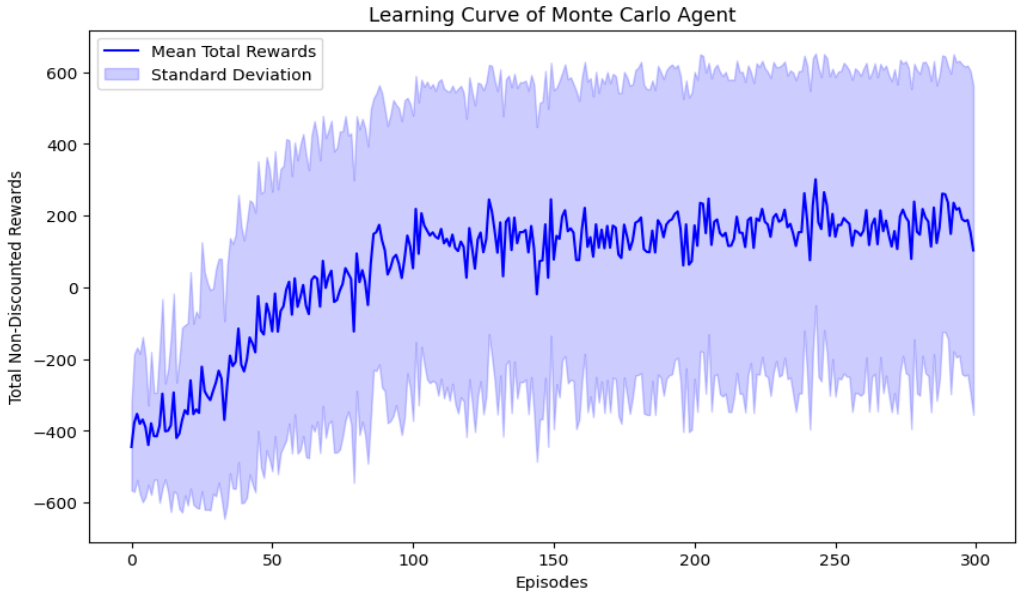

Figure 5: Learning curve of Monte-Carlo agent.

(d) Consider an implementation which ignores trajectories that don't reach terminal states within 500 steps. In what ways would this affect the MC estimate of the value function compared to an implementation that does not impose a step limit per episode?

If we ignored trajectories that do not reach terminal states, the value function would increase for states that take part in these trajectories. Particularly those that are not close to a terminal state would suffer a grater increase. This occurs because, in these types of trajectories, the accumulated non-discounted reward is always negative (-500 for a non-terminating trajectory). As a result, many states would receive an approximate value corresponding to this negative reward. By ignoring these trajectories, we are effectively avoiding the negative rewards, which leads to an increase in the value.

3. Temporal Difference Reinforcement Learning

(a) We have chosen the Q-learning method to solve this problem. In this case, we needed to set the learning rate (α), ϵ (for the ϵ-greedy policy), and the number of episodes. As with the Monte Carlo approach, we observed in the agent's learning curve that, after approximately 250 episodes, it no longer showed improvement. Therefore, we set the number of episodes to this value. For ϵ, we found empirically that high exploration levels did not yield the desired results in this case. Thus, we chose a faster decay rate:

$$\epsilon = \max(0.2, \epsilon \times 0.9)$$

which provided better performance. This approach is reasonable, as Q-learning typically depends less on exploration than our Monte Carlo agent due to the nature of the algorithm. Finally, we set $\alpha = 0.1$, a commonly used value that meets the requirement of being between 0 and 1. Empirical observations suggest that it allows for stable and progressive learning. It provides a good balance between the speed of learning and the stability of updates, promoting more reliable convergence.

(b)

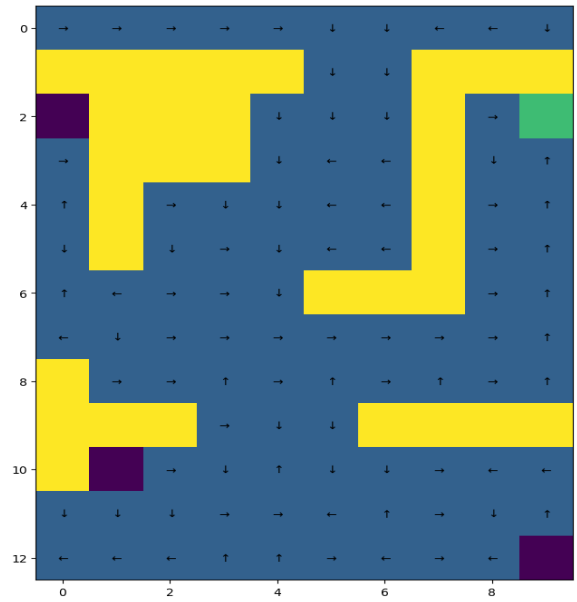

Figure 6: Optimal policy of Temporal Difference.

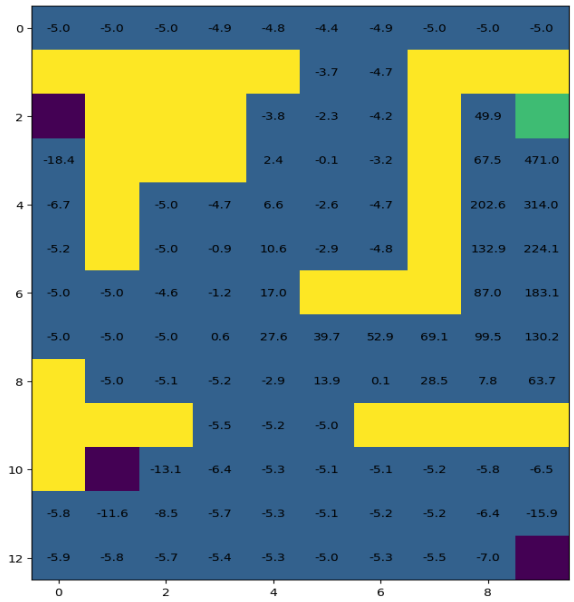

Figure 7: Optimal value of Temporal Difference.

(c)

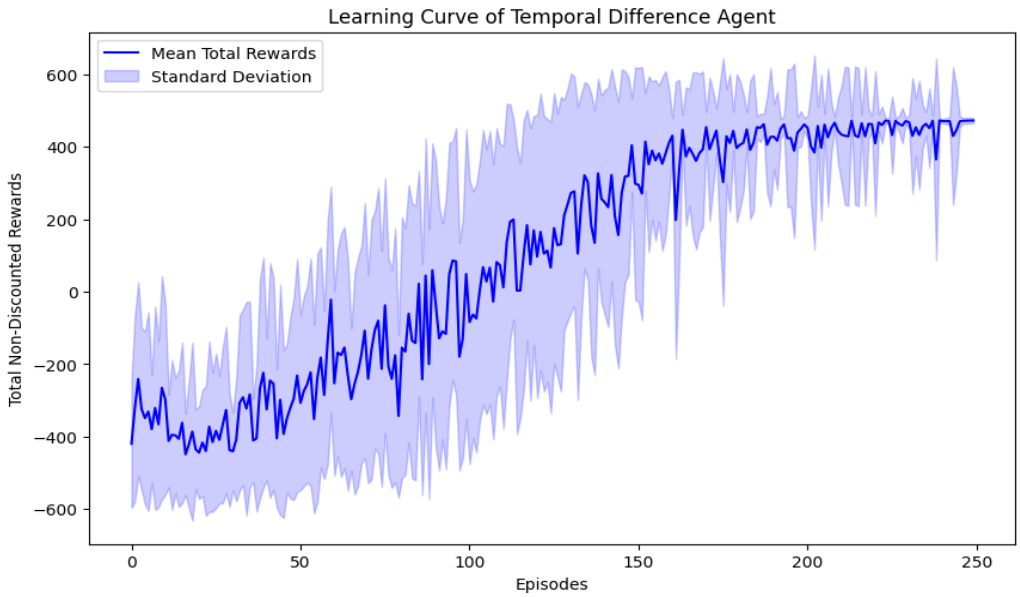

Figure 8: Learning curve of Temporal Difference agent.

(d) Do off-policy algorithms need to be greedy in the limit with infinite exploration (GLIE), in order to guarantee that the optimal action-value function is learned?

GLIE is not a requisit for off-policy methods because the behaviour policy can be more exploratory, even while the target policy is improving. As long as the behaviour policy explores the state space sufficiently, the off-policy algorithm will converge to the optimal policy.