Implementation of Q-Learning Algorithm

Ahaj Mahhin Faiak
Department of Computer Science and Engineering
University of Dhaka
ahaj-2020215611@cs.du.ac.bd

1 Introduction

Reinforcement Learning (RL) is a branch of machine learning concerned with how agents ought to take actions in an environment so as to maximize cumulative rewards. Among the various RL techniques, Q-Learning is one of the most widely used due to its simplicity, model-free nature, and ability to learn optimal policies through trial and error.

Q-Learning is an off-policy, value-based method that seeks to learn the optimal action-value function, denoted as $Q(s, a)$, which represents the expected utility of taking action a in state s and following the optimal policy thereafter. The algorithm iteratively updates the Q-values using the Bellman optimality equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r(s) + \gamma \max_{a'} Q(s', a') - Q(s, a) \right],$$

where α is the learning rate, γ is the discount factor, $r(s)$ is the immediate reward, and s' is the next state.

The primary objective of this work is to implement the Q-Learning algorithm in a simulated environment, analyze the learned policies, and visualize the agent's traversal and value function through heatmaps. This report details the design of the environment, the algorithm's parameters, and the experimental results obtained from multiple training runs.

2 Problem Formulation

The environment is defined as a discrete two-dimensional grid of size $m \times n$. The agent begins at the initial state $(0, 0)$ and must navigate to a goal state (x_g, y_g), which is randomly selected at the start of each experiment. A set of obstacle states is also randomly placed within the grid; these states are inaccessible and yield a large negative reward if entered.

At each time step, the agent can choose one of four actions: `up`, `down`, `left`, or `right`. The objective is to learn a policy π that selects actions so as to maximize the cumulative discounted reward, which is achieved by reaching the goal state while avoiding obstacles. The episode terminates when the agent reaches the goal state or exceeds a predefined maximum number of steps.

3 Implementation

The agent is trained to navigate a randomly generated grid environment, starting from the top-left corner $(0, 0)$ and aiming to reach a randomly placed goal. Obstacles are also randomly

scattered throughout the grid, representing impassable cells. The agent can move in four directions: up, down, left, and right, but cannot move outside the grid boundaries or into obstacles.

During training, the agent receives a large positive reward for reaching the goal and a significant negative penalty for hitting an obstacle. For other moves, the reward is zero. The agent learns an optimal policy by balancing exploration and exploitation, gradually improving its action choices through repeated episodes.

After training, the agent's learned policy is used to determine a path from the start to the goal, which is visualized alongside a heatmap representing the learned value estimates for each grid cell. This process is repeated multiple times to analyze performance across different random environments.

The learning rate (α) is set to 0.1, the exploration rate (epsilon, ε) is fixed at 0.1, and the discount factor (γ) is maintained at 0.9 across all environments.

3.1 Results and Analysis

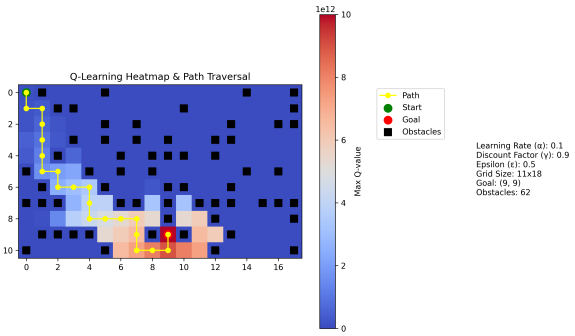

Figure 1: Grid environment 1

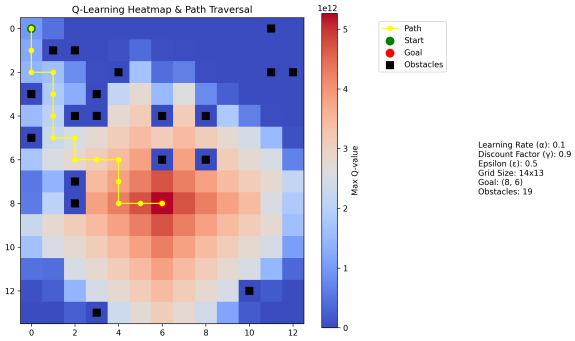

Figure 2: Grid environment 2

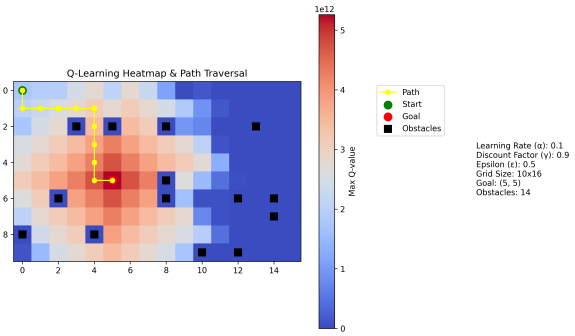

Figure 3: Grid environment 3

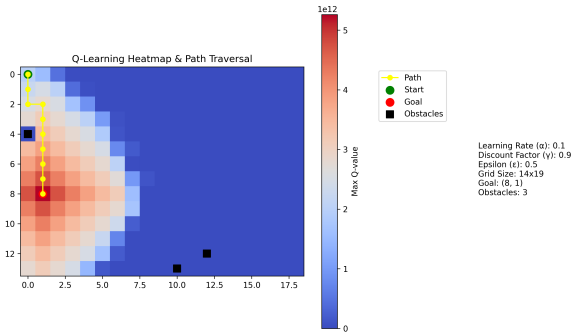

Figure 4: Grid environment 4

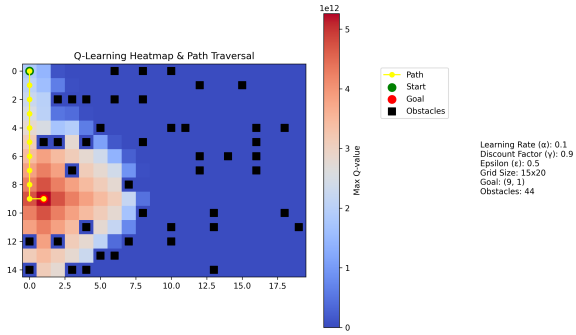

Figure 5: Grid environment 5

The heatmaps demonstrate that Q-learning is an efficient algorithm, as it effectively focuses exploration on relevant states while avoiding unnecessary ones. This highlights that properly tuning the parameters can significantly enhance the algorithm's performance.

4 Conclusion

This work presented the implementation of the Q-learning algorithm in a grid-based environment with randomly placed goals and obstacles. The results show that the agent successfully learns to navigate towards the goal while avoiding obstacles, demonstrating the effectiveness of Q-learning for discrete pathfinding problems. Proper tuning of learning parameters such as learning rate, discount factor, and exploration rate is essential for achieving good performance.