## Q-Learning

When deciding on how to move the robot in a specific direction and updating the environment's state. It involves:

1. Initializing the robot's current position and score
2. Checking if the move direction is within grid boundaries and is not into a wall
3. Updating the robot's position based on the direction
4. Adding a visit penalty for revisiting cells
5. Rewarding or Penalizing the robot based on the cell type of delivery, high-traffic, or terminal state
6. Clears the cell after the robot moves
7. Updating the score with the computed reward
8. We also ensure that the score remains finite by resetting it to `initial_score` if necessary.

Hence we can explore new areas and avoid penalties based on cell type of delivery, high-traffic, or terminal state.

## Q-Learning with ε-Greedy

Here we are trying to navigate the grid world by balancing exploration and exploitation. The function, we:

1. Start with a random start position
2. On choosing the next action:
    a. With probability ε (epsilon), it explores and the robot selects a random action to explore new states.
    b. With probability `1-ε`, the robot selects the action with the highest Q-value to exploit known information.
3. It executes the selected action using the `move` function, which updates the robot's position and computes the reward.
4. If the new position is out of bounds or a wall, the reward is penalized, and the robot is reset to a random position.
5. It updates the Q-value using the Q-learning update rule.
6. The visit count is increased for the state-action pair.
7. The screen update refreshes the visual representation of the grid world.
8. The `decay `k` gradually reduces the value of `k` to shift from exploration to exploitation over time.

## Q-Learning with Exploration Function

Here we are trying to navigate the grid world optimally using Q-learning with exploration bonuses. In the function, we:

1. Start with a random start position

2. Use the exploration function
    a. During exploration, we randomly select an action based on a parameter `k`.
    b. During exploitation, we use the Q-values and visit counts to decide the best action.
3. It moves based on the selected action and calculates the reward.
4. The Q-value is updated based on the received reward and the maximum future Q-value.
5. The visit count is increased to track exploration.
6. The screen update refreshes the visual representation of the grid world.
7. The `decay `k` gradually reduces the value of `k` to shift from exploration to exploitation over time.

## Parameter Tuning

**a. Exploration Rate with ε-Greedy Analysis**

| Epsilon Value (ε) | Episodes Tested | Cumulative Rewards (Range) | Average of Final Scores | Observations |
|---|---|---|---|---|
| 1.0 | 13 | 34.0 to -176 | -56.0 | Full exploration; slow convergence, poor performance. |
| 0.9 | 5 | -34.0 to -152 | -116.4 | Mix of exploration and exploitation; inconsistent rewards. |
| 0.7 | 5 | -8.0 to -176 | -117.2 | Faster convergence; more exploitation, better performance. |
| 0.5 | 6 | -8.0 to -170 | -125.00 | Balanced exploration; stabilized rewards but worse performance. |
| 0.3 | 7 | -2.0 to -200 | -139.14 | Less exploration; faster stabilization, moderate performance. |
| 0.1 | 4 | -74.0 to -176 | -146.0 | Minimal exploration; heavy exploitation, faster convergence. |

**Learning Rate and Discount Factor Analysis**

Experiments were conducted by varying the **learning rate (α)** and **discount factor (γ)**. The impact of these parameters on performance is summarized below:

| **Parameter** | **Values Tested** | **Observations** |
|---|---|---|
| **Learning Rate (α)** | 0.1, 0.5, 0.9 | Higher α (e.g., 0.9) allowed faster convergence but risked overshooting optimal policies. Lower α slowed learning. |
| **Discount Factor (γ)** | 0.5, 0.7, 0.9 | Higher γ encouraged long-term rewards but delayed convergence. Moderate γ (e.g., 0.7) provided a good balance. |

**b. Exploration Parameter k Analysis**

Experiments were conducted with various values of exploration function's parameter k to observe its impact on learning efficiency and agent behavior. Results are summarized below:

| **k** | **Observations** | **Final Score** |
|---|---|---|
| 0.5 | Encouraged balanced exploration. Converged after moderate exploration with optimal path discovery. | -212.0 |
| 1.0 | Enhanced exploration but slowed convergence due to overemphasis on less-visited states. | 16.0 |
| 1.5 | Excessive exploration hindered optimal policy learning, leading to higher penalties. | -818.0 |

## Convergence Analysis

### Episodes to Learn Good Policies

The convergence rates for the two Q-learning methods were analyzed based on the number of episodes required to stabilize cumulative rewards and achieve consistent optimal performance:

| **Method** | **Episodes to Converge** | **Observations** |
|---|---|---|
| **ε-Greedy** | 20–30 | Often suboptimal due to limited exploration |

| | | |
|---|---|---|
| **Exploration Function** | 30–50 | Better long-term performance due to thorough exploration |

**Effects of ε-Greedy Strategy:**

- Early high exploration enabled the agent to explore diverse states.
- Gradual reduction in ε improved exploitation but limited discovery of less-visited states.

**Effects of Exploration Function Strategy:**

- Enhances exploration by adding a bonus reward to encourage the agent to visit less-explored states, preventing them from getting stuck in local optima.
- Enhances long-term performance by allowing agents to discover optimal policies over time, despite initial slower convergence, as they explore a wider range of states and actions.

## Performance Comparison

| Aspect | ε-Greedy | Exploration Function | Observations |
|---|---|---|---|
| **Convergence Speed** | Faster (20–30 episodes) | Slower (30–50 episodes) | ε-Greedy converged faster but less robustly. |
| **Cumulative Rewards** | Moderate | Higher | Exploration Function achieved higher long-term rewards. |
| **State Coverage** | Limited | Extensive | Exploration Function explored less-visited states more effectively. |
| **Learning Stability** | Medium | High | Exploration Function produced consistent policies. |

**Better Performing Approach**

- **ε-Greedy Approach:**
    - Suitable for faster learning where immediate performance is significant.
    - Simpler implementation with quicker convergence but less robust.
- **Exploration Function Approach:**

- Achieved higher cumulative rewards and better state coverage.
- Produced more stable policies and improved overall learning performance.
- Ideal for scenarios requiring extensive state exploration.

## Conclusion

For Q-learning with ε-Greedy, the optimal performance was achieved with a learning rate of $\alpha = 0.9$, discount factor $\gamma = 0.9$, and exploration rate = 1.0.

For Q-learning with a bonus reward, 1.0 is the optimal performance for the parameter k of the exploration function.

The Q-Learning with Exploration Function performed better than Q-Learning with ε-Greedy after continuous evaluation, attaining faster convergence and bigger cumulative rewards.