**1. Q-Learning Method**

**Overview**:  
Q-learning is a reinforcement learning algorithm where an agent learns the optimal action-selection policy by interacting with its environment. It updates a Q-value for each state-action pair based on rewards received.

**Key Parameters:**

* **Alpha (Learning Rate)**: Determines how much of the new information overrides the old. Here, alpha = 0.1.
* **Gamma (Discount Factor)**: Determines the importance of future rewards. gamma = 0.9 emphasizes both immediate and future rewards.
* **Epsilon (Exploration Rate)**: Controls exploration vs. exploitation. epsilon = 0.1 allows 10% of actions to be random for exploration.

**Q-value Update Formula:**

Q(s,a)←Q(s,a)+α[r+γmax⁡aQ(s′,a)−Q(s,a)]Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max\_a Q(s', a) - Q(s, a) \right]Q(s,a)←Q(s,a)+α[r+γamax​Q(s′,a)−Q(s,a)]

Where:

* sss: Current state.
* aaa: Chosen action.
* rrr: Reward received.
* s′s's′: Next state.

**Implementation in the Code:**

* **State Representation**: Number of vehicles on lanes controlled by the traffic light.
* **Action**: Selecting a traffic light phase.
* **Q-table**: A dictionary storing Q-values for state-action pairs.

**2. Reward System**

The reward system balances throughput, waiting times, congestion, and phase duration.

**Reward Components:**

1. **Reward for Waiting Time**:

Reward\_waiting=−Total Waiting Time\text{Reward\\_waiting} = -\text{Total Waiting Time}Reward\_waiting=−Total Waiting Time

* + Penalizes prolonged vehicle waiting.

1. **Reward for Throughput**:

Reward\_throughput=k1×Throughput\text{Reward\\_throughput} = k\_1 \times \text{Throughput}Reward\_throughput=k1​×Throughput

* Incentivizes clearing vehicles through intersections. Here, k1=2.0k\_1 = 2.0k1​=2.0.

1. **Penalty for Congestion**:

Penalty\_congestion=−k2×max⁡(0,Queue Length−Qthreshold)\text{Penalty\\_congestion} = -k\_2 \times \max(0, \text{Queue Length} - Q\_{\text{threshold}})Penalty\_congestion=−k2​×max(0,Queue Length−Qthreshold​)

* Penalizes long queues beyond a threshold. k2=0.5k\_2 = 0.5k2​=0.5, Qthreshold=5Q\_{\text{threshold}} = 5Qthreshold​=5.

1. **Penalty for Long Phase Durations**:

Penalty\_long\_phase=−k3×max⁡(0,Phase Duration−Pthreshold)\text{Penalty\\_long\\_phase} = -k\_3 \times \max(0, \text{Phase Duration} - P\_{\text{threshold}})Penalty\_long\_phase=−k3​×max(0,Phase Duration−Pthreshold​)

* + Discourages phases that are too long. k3=0.2k\_3 = 0.2k3​=0.2, Pthreshold=10P\_{\text{threshold}} = 10Pthreshold​=10.

**Total Reward:**

Total Reward=Reward\_waiting+Reward\_throughput+Penalty\_congestion+Penalty\_long\_phase\text{Total Reward} = \text{Reward\\_waiting} + \text{Reward\\_throughput} + \text{Penalty\\_congestion} + \text{Penalty\\_long\\_phase}Total Reward=Reward\_waiting+Reward\_throughput+Penalty\_congestion+Penalty\_long\_phase

**3. Action Selection (Epsilon-Greedy Policy)**

The choose\_action function uses an epsilon-greedy approach:

* **Exploration**: With a probability of ϵ=0.1\epsilon = 0.1ϵ=0.1, selects a random action.
* **Exploitation**: With a probability of 1−ϵ1 - \epsilon1−ϵ, selects the action with the highest Q-value.

This strategy ensures a balance between discovering new strategies and leveraging learned ones.

**4. Simulation Workflow**

1. **Initialization**:
   * The SUMO simulation starts with traci.start.
   * Traffic light ID is obtained using traci.trafficlight.getIDList().
2. **Training Loop**:
   * For each step:
     + **State**: Extracted from the number of vehicles on lanes.
     + **Action**: Selected using the Q-table or random choice.
     + **Reward**: Calculated using the reward system.
     + **Q-table Update**: Performed using the Q-learning formula.
3. **Evaluation**:
   * After training, the model is evaluated over 500 steps to calculate average waiting time and rewards.

**5. Visualizing Training Progress**

The code tracks:

* **Steps Progress**: The simulation steps.
* **Rewards at Steps**: Rewards obtained at each step.

A plot is generated to show reward trends during training, providing insights into the agent's learning process.