Ryan Aden George

CIA2-RL

**Problem Definition**

1. **Environment**: A 100x100 grid with two randomly selected points as the start and goal positions.
2. **Obstacles**: Randomly placed cells in the grid that act as impassable obstacles, except for the start and goal positions.
3. **Agent Objective**: To find the optimal path from the start to the goal while avoiding obstacles, minimizing movement costs, and maximizing the rewards.

**Step 1: Define the MDP (Markov Decision Process)**

1. **States**: Each cell in the 100x100 grid represents a unique state.
2. **Actions**: The agent can take one of four possible actions at each state:
   * Move **up**
   * Move **down**
   * Move **left**
   * Move **right**
3. **Transition Model**:
   * Each action moves the agent to an adjacent cell.
   * If an action leads to an obstacle or off the grid, the agent remains in the current state.
4. **Rewards**:
   * **Goal state**: A large positive reward.
   * **Each step**: A small negative reward to encourage the shortest path.
   * **Obstacle hit**: A large negative reward to avoid obstacles.

### Step 2: Implement Different RL Algorithms

1. **Dynamic Programming (DP)**:
   * Use **Value Iteration** or **Policy Iteration**.
   * Requires a known transition and reward model.
   * Provides a baseline by computing an optimal policy for the grid-based MDP using iterative updates.
2. **Q-learning (Off-Policy, Model-Free RL)**:
   * Initialize a Q-table for each state-action pair.
   * Use the **epsilon-greedy** strategy to balance exploration and exploitation.
   * Update the Q-values iteratively based on the agent’s experience using the **Bellman equation**.

3. **Actor-Critic Method (Policy-Based, Model-Free RL)**:

* + Employs two networks: one for policy (actor) and one for value estimation (critic).
  + The actor network decides actions, while the critic network evaluates the value of the state-action pair.
  + Useful for continuous or large state-action spaces.

### Step 3: Benchmarking and Evaluation

1. **Performance Metrics**:
   * **Total Rewards**: Sum of rewards accumulated by the agent to reach the goal.
   * **Convergence Rate**: Number of episodes required for the agent to reach stable policies and rewards.
   * **Path Efficiency**: Number of steps taken to reach the goal from the start.
2. **Benchmark Setup**:
   * Run each algorithm (DP, Q-learning, Actor-Critic) in the grid environment.
   * Record performance metrics for each algorithm and evaluate their effectiveness in terms of policy optimality and action efficiency.
   * Measure computation time for each method to evaluate the algorithm's computational efficiency.
3. **Result Comparison**:
   * Compare each algorithm's efficiency in finding an optimal or near-optimal policy.
   * **DP** (baseline) is expected to provide the optimal policy but may be computationally intense.
   * **Q-learning** should perform well with appropriate exploration parameters.
   * **Actor-Critic** is expected to perform better in larger environments but may take longer to converge.

**Step 4: Visualization**

1. **Path Visualization**:
   * Plot the paths taken by each agent on the grid.
   * Visualize start and goal points, obstacles, and the path trajectory for each algorithm.
2. **Performance Plotting**:
   * Plot convergence curves for each algorithm, showing the improvement in total rewards over episodes.
   * Visualize path efficiency by plotting the number of steps taken across episodes.

This approach provides a comprehensive framework to test and compare RL methods for a grid-based navigation problem with obstacles.

**Diagram 1: Grid Environment Setup**

+--------------------------------------------------------------------+

| 100x100 Grid |

|--------------------------------------------------------------------|

| Start Point Obstacles (Random) Goal Point |

| |

| Randomly placed obstacles Random |

| across grid, not on start/goal. end location |

|  |
| --- |
|  |

 **Start and Goal Points**: Marked on opposite sides of the grid.

 **Obstacles**: Randomly placed cells representing inaccessible locations.

**Diagram 2: MDP Definition**

+-------------------- MDP for Grid ------------------+

| State |

| (Each cell in grid) |

|------------------------|---------------------------|

| Actions | Transition Model |

| Move up, down, left, | Action → Next State |

| right | (blocked by boundaries |

| | and obstacles) |

|------------------------|---------------------------|

| Rewards | Goal (+), Obstacles (-) |

| -1 for each move, | Steps have penalties |

| +10 for goal, -10 for | Goal is high reward |

| obstacle | |

+----------------------------------------------------+

* **States**: Represent grid cells.
* **Actions**: Four possible movements.
* **Transition Model**: Defines movement and obstacle boundaries.
* **Rewards**: Encourage shortest path while avoiding obstacles.

### Step 2: Implement RL Algorithms

**Diagram 3: RL Algorithms Overview**

+------------------ RL Algorithms for MDP -----------------+

| |

| 1. Dynamic Programming |

| (Value Iteration / Policy Iteration) |

| - Uses known transition and reward models |

| - Computes optimal policy through updates |

|---------------------------------------------------------|

| |

| 2. Q-learning |

| - Model-free, uses Q-table |

| - Epsilon-greedy for exploration |

| - Updates Q-values using experience |

|  |
| --- |
|  |

**Step 3: Benchmarking and Evaluation**

**Diagram 4: Performance Metrics and Benchmarking**

+------------------ Performance Evaluation ---------------------+

| |

| 1. Metrics |

| - Total Rewards (sum of all steps) |

| - Convergence Rate (stabilization of rewards) |

| - Path Efficiency (steps taken to goal) |

|---------------------------------------------------------------|

| |

| 2. Benchmark Setup |

| - Run algorithms multiple times |

| - Track and record metrics for each |

|---------------------------------------------------------------|

| |

| 3. Result Comparison |

| - Compare against baseline (DP) |

| - Evaluate other algorithms for policy optimality |

| - Measure computational efficiency |

+---------------------------------------------------------------+

* **Total Rewards**: Measures accumulated rewards per episode.
* **Convergence Rate**: Tracks episodes until stable performance.
* **Path Efficiency**: Counts steps taken to reach the goal.