**Analysis Report**

**Introduction**

**Objectives of the Project**

The primary objective of this project is to implement and analyze a model-free reinforcement learning algorithm known as **Q-learning** in a grid-based multi-agent system. The simulation aims to enable agents to optimize their behaviors through interacting with their environment and improving their overall decision-making over time. Key tasks from this project include:

* Developing a Q-learning algorithm to train a single agent to navigate a grid-world environment efficiently.
* Expanding the environment to a multi-agent setup and studying the interactions between the agents.
* Evaluating agent performance through different learning curves that focus on the total rewards per episode and the episode length.

**Overview of Q-learning and Reinforcement Learning**

**Reinforcement Learning (RL)** is a branch of machine learning where agents learn optimal actions through interacting with their environment. Unlike supervised learning, reinforcement learning does not rely on labeled data for training the agents, but instead it relies on penalties and rewards to guide the learning process. These penalties and rewards are defined depending on the problem at hand and the goal we are trying to achieve. The agent’s goal is to maximize the cumulative reward over time by learning an optimal policy—a mapping of states to actions.

There are two different branches of reinforcementlearning**:** model-based and model free. Model-based algorithms include policy iteration and value iteration. This project utilized a model-free algorithm known as **Q-learning.** This is a model-free RL algorithm that allows the agents to learn optimal policies for decision-making tasks. It uses a tabular approach, maintaining a **Q-table** which maps state-action pairs to estimated future rewards. At each step:

1. The agent selects an action based on its current policy (an epsilon-greedy approach was the policy utilized in this project, more on that later).
2. It observes the environment’s response in the form of a reward and a new state. As outlined in this project, any movement by the agent would give it a -1 penalty unless it arrived at the goal state which would then be a +20 reward.
3. Once the agent knows the resulting state and its immediate reward, it updates the Q-value for this state-action pair using the following equation:

**Q(s, a) 🡨 Q(s, a) + α (r + γ max Q(s’, a’) – Q(s, a) )**

Here:

* + - **Q(s, a)**: The Q value for state s and action a.
    - **r**: Immediate reward received
    - **γ**: Discount factor, controlling the importance of future rewards
    - **α**: Learning rate, determining how much the new information updates the current Q-value.
* This iterative process outlined right above allows the agent to improve its decision-making and converge to an optimal policy.

**Q-Learning Parameters**

The success of the Q-learning algorithm relies heavily on a few hyperparameters that we need to carefully choose. These hyperparameters determine how an agent balances between exploration and exploitation, learning, and prioritization of immediate vs. future rewards. Below is a brief explanation of these key parameters used in the project and their impact on learning:

1. **Learning Rate (α)**

The learning rate determines the extent to which newly acquired information overrides the existing Q-values during updates.

As mentioned above, the Q-value update rule is as follows:

**Q(s, a) 🡨 Q(s, a) + α (r + γ max Q(s’, a’) – Q(s, a) )**

* **α** represents the learning rate and is a value between 0 and 1.

**Impact on Learning:**

* **A high learning rate** (e.g α = 0.9):
  + Makes the agent adapt quickly to new information.
  + Risks instability as it may overfit to recent experiences and discard useful past knowledge.
* **A low learning rate** (e.g α = 0.1):
  + Ensures smoother, more stable learning.
  + May result in slower convergence to the goal state.

**Chosen Value:**

* For this project, a learning rate **α** was set to 0.5, striking a balance between adapting to new information and stable learning.

1. **Discount Factor (γ)**

The **discount factor (γ)** determines the importance of future rewards to immediate rewards.

The Q-value update rule uses:

**r + γ maxQ(s′, a′)**

* **γ**: A value between 0 and 1.

**Impact on Learning:**

* A **high discount factor** (e.g., γ=0.9):
  + Prioritize long-term rewards over immediate rewards.
  + Helps the agent focus on strategic, goal-oriented behavior.
* A **low discount factor** (e.g., γ=0.1):
  + Emphasizes short-term rewards, making the agent myopic.
  + Risks suboptimal behavior in tasks requiring planning.

**Chosen Value:**

For this project, γ was set to 0.9, reflecting the importance of long-term rewards, as the goal is to reach a distant target while minimizing penalties for unnecessary moves.

1. **Exploration Rate (ϵ)**

The **exploration rate (ϵ)** controls the trade-off between **exploration** (trying new actions) and **exploitation** (choosing the best-known action).

An **epsilon-greedy strategy** is used:

* With probability ϵ, the agent chooses a random action (exploration).
* With probability 1−ϵ, it chooses the action with the highest Q-value (exploitation).

**Impact on Learning:**

* A **high exploration rate** (e.g., ϵ=0.5):
  + Ensures the agent explores diverse actions, reducing the risk of getting stuck in local optima.
  + May result in slower convergence, as the agent spends more time exploring.
* A **low exploration rate** (e.g., ϵ=0.1):
  + Accelerates convergence by focusing on exploitation.
  + Risks suboptimal solutions if exploration is insufficient.

**Chosen Value:**

The initial ϵ was set to **0.1**, encouraging exploration at the start. An epsilon decaying strategy was used where ϵ decays gradually over time to a minimum of 0.01, ensuring the agent shifts focus to exploitation as it learns.

**Impact of Parameter Choices**

The chosen parameters ensure:

1. **Stable Learning**:
   * A moderate learning rate (α=0.5) balances adaptation and stability.
2. **Strategic Behavior**:
   * A high discount factor (γ=0.9) helps the agent focus on long-term goals.
3. **Efficient Exploration**:
   * The epsilon-decay strategy allows the agent to explore early on and exploit its knowledge later.

These parameters were tuned to optimize performance in the grid environment, ensuring the agent learns to navigate efficiently to the goal while minimizing penalties.

**Single-Agent Behavior Analysis**

**Observations**

In the **Total** **Reward** vs. **Episode** plots:

* Early episodes usually have lower rewards due to more exploration being done in those early stages of the experiment.
* As the episodes progress, the rewards generally increase, showing the agent is learning to reach the goal more efficiently.

In the **Episode** **Length** vs. **Episode** plots:

* Initial episodes are often longer as the agent’s actions are geared more towards exploration than exploitation.
* Over time, the lengths of the episodes decrease, indicating that the agent is finding shorter paths to the goal and favoring exploitation of optimal actions over random exploration.

**Learning Curves**

A graph with orange line

Description automatically generated**A graph of a graph

Description automatically generated**

**Multi-Agent Behavior Analysis**

**1. Learning Performance**

**Observations**:

* **200 Episodes:**
  + Early episodes show low rewards and high steps for all agents. This indicates that agents are still exploring the environment and have not learned optimal paths. This is due to the epsilon-greedy we have used where exploration is favored in the beginning stages of the experiment.
  + Over time, rewards begin to increase, and steps decrease, suggesting agents are converging toward better strategies. This is because the agents favor exploitation over exploration as the epsilon decays over the episodes.
* **800 Episodes:**
  + Rewards are significantly higher than in the 200-episode run, showing improved efficiency.
  + Steps per episode are more consistent, with most agents converging on shorter paths.
* **1000 Episodes:**
  + Rewards plateau, and steps stabilize, indicating the agents have largely learned the optimal paths to the goal.
  + Variability in rewards between agents reduces, suggesting more consistent performance.

**2. Collaboration vs. Competition**

**Observations:**

* **Collaboration**:
  + Agents sharing a Q-table may have shown better collaborative behavior (e.g., avoiding redundant paths).
* **Competition**:
  + In some episodes, agents compete for the same paths, leading to inefficient learning for one or more agents.

**Examples from the data:**

* **200 Episodes**:
  + Some episodes have agents with highly negative rewards, suggesting interference or suboptimal strategies.

A graph with orange line

Description automatically generatedA graph with colorful lines

Description automatically generated

* **800 and 1000 Episodes**:
  + Collaboration becomes more evident as agents learn to avoid obstacles and each other, reducing interference.

A graph with numbers and lines

Description automatically generatedA graph with numbers and lines

Description automatically generated

A graph with orange lines

Description automatically generatedA graph with orange lines

Description automatically generatedA graph with orange lines

Description automatically generated

**3. Observations and Patterns**

1. **Agent-Specific Performance**:
   * Agent 1 often has higher or more consistent rewards than others, possibly due to better initial positioning or fewer conflicts throughout the simulation.
   * Agents starting closer to the goal tend to perform better in early episodes.
2. **Exploration vs. Exploitation**:
   * Early episodes show high variability in rewards and steps due to exploration.
   * Later episodes reflect more exploitation of learned strategies, with reduced variability.
3. **Interference**:
   * In some episodes, rewards for one or more agents are significantly lower, suggesting they were blocked or forced into suboptimal paths.

**Conclusion**

* **Learning Efficiency**:
  + The agents improve their performance over time, with rewards stabilizing and steps decreasing.
* **Collaboration vs. Competition**:
  + Early episodes show significant competition and interference, but this reduces in later episodes as agents converge on optimal strategies.
* **Agent Behavior**:
  + Agents with independent Q-tables take longer to converge, whereas shared Q-tables may have promoted better collaboration but could have also caused initial instability.