**Reinforced Multiagent Learning System in a Transport World**

# Design Overview

The project involves developing a reinforced multiagent learning system in a transport world, that navigates a grid environment, aiming to reach a target location while avoiding obstacles. The system uses reinforcement learning (Q-learning) to train 3 agents to make optimal decisions.

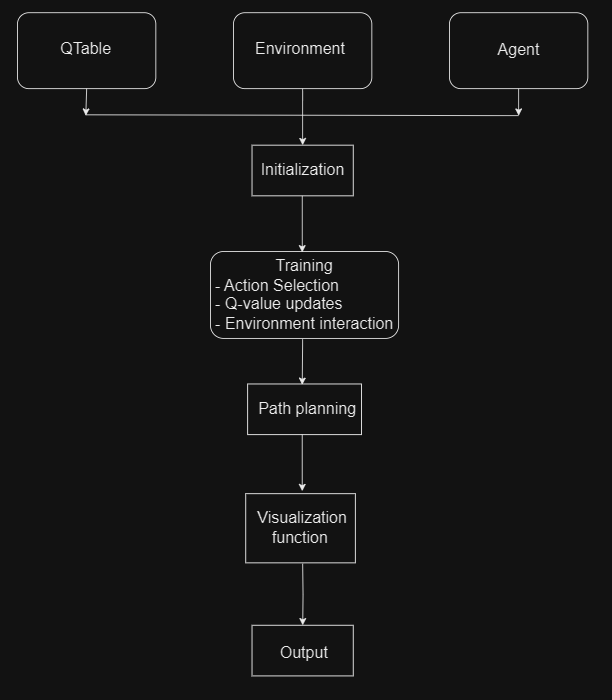
# Classes

* QTable: Represents the Q-table used for Q-learning. It stores Q-values for each state-action pair and provides methods to update and access these values.
* Environment: Represents the grid environment in which the agent operates. It includes methods to define the environment size, rewards for each state, and terminal states.
* Agent: Represents the AI agent that learns to navigate the environment. It uses the Q-table to make decisions and update Q-values during training.
* Multiple helper and visualization funtions.

# Interactions

* Initialization: The QTable and Environment classes are initialized with the grid size and reward structure. The Agent class is initialized with the QTable and environment.
* Training: The Agent class runs through multiple training episodes, during which it selects actions based on epsilon-greedy strategy, updates Q-values using the Q-learning algorithm, and moves to the next state.
* Path Planning: After training, the Agent class can determine the shortest path from any starting position to the target location using the learned Q-values.
* Visualization: Various visualization functions are used to display the environment, agent's path, and Q-table.

# Workflow Diagram



# State Space Design

The state space in this project represents all possible states the agent can be in within the grid environment. Each state is defined by the agent's current position in the grid. The state space is represented by a 2D grid, where each cell corresponds to a possible state. In this case, the state space is a 5x5 grid.

# Q-Table Design

The Q-table is a data structure used by the agent to store the Q-values for each state-action pair. It is a 3D array with dimensions corresponding to the number of rows, columns, and possible actions (North, East, South, West, Pickup, Dropoff). Initially, all Q-values are set to 0.

# Implementation of Constraints

* Environment Constraints: The grid environment contains obstacles and terminal states (item packaging area and item storage locations). These are represented by rewards in the rewards matrix. The agent cannot move to obstacle states and must avoid them.
* Agent Constraints: The agent follows an epsilon-greedy policy for action selection, where it chooses the action with the highest Q-value with probability 1-epsilon and a random action with probability epsilon for exploration.
* Q-Learning Constraints: During training, the agent updates the Q-values using the Q-learning algorithm, considering the reward received for the action taken and the maximum Q-value for the next state.
* Path Planning Constraints: After training, the agent can find the shortest path from any starting position to the item packaging area or item storage locations using the learned Q-values.

# Experiments

## Experiment 1

### Experiment Summary:

The experiment aims to evaluate the performance of three agents (red, blue, and black) in a grid world environment using Q-learning. The agents are trained to navigate the grid and maximize their rewards by following different policies (random, greedy, and exploit). The experiment compares the agents' performance under these policies over 10,000 steps.

Pseudocode:

1. Initialize environment parameters (rows, cols, actions, learning\_rate, discount\_factor).
2. Initialize agents with Q-tables based on the environment parameters.
3. Set initial environment state and choose a policy (PRANDOM).
4. Run 500 steps with the random policy.
5. Print results for the first run.
6. Reset the environment state and run 8500 steps with the random policy.
7. Print results for the second run.
8. Reset the environment state and switch the policy to PGREEDY.
9. Run 8500 steps with the greedy policy.
10. Print results for the greedy run.
11. Reset the environment state and switch the policy to PEXPLOIT.
12. Run 8500 steps with the exploit policy.
13. Print results for the exploit run.
14. Print the final Q-tables of each agent.

Steps Explanation:

* Environment Initialization: The grid world environment is set up with 5 rows and 5 columns. There are 6 possible actions (North, East, South, West, Pick, Drop) that the agents can take. The learning rate is set to 0.3, and the discount factor is set to 0.5.
* Agent Initialization: Three agents (redAgent, blueAgent, blackAgent) are initialized with their respective Q-tables based on the environment parameters.
* **Running the Experiment:**
* First 500 Steps (Random Policy): Each agent takes turns to move randomly in the environment for 500 steps. If a terminal state is reached, the environment is reset.
* Second Round (Greedy Policy): Agents switch to a greedy policy where they choose actions that lead to the highest Q-value in each state. They run for 8500 steps.
* Third Round (Exploit Policy): Agents switch to an exploit policy where they choose actions based on the highest Q-value with a high probability (0.95) and randomly with a low probability (0.05). They run for 8500 steps. This step helps in balancing exploration and exploitation.
* **Results Demonstration:**
* Random Run (500 steps): The agents' initial performance under a random policy is observed. This helps in understanding the baseline behavior of the agents.
* Greedy Run (8500 steps): The agents learn to exploit the environment more effectively, resulting in higher rewards than the random run.
* Exploit Run (8500 steps): The agents continue to exploit the environment but with occasional exploration, resulting in slightly different rewards than the greedy run.

### Results Explanation:

**First Random Run (500 steps):**

* Objective: Evaluate initial performance and behavior.
* Outcome: The agents' performance is poor, as expected, due to the random nature of the policy. They do not learn optimal strategies and incur negative rewards.

**Second Run (8500 steps):**

* Objective: Assess learning progress.
* Outcome: The agents' performance improves significantly compared to the first run. They start to learn better strategies and achieve higher rewards, indicating learning from the environment.

**Greedy Run (8500 steps):**

* Objective: Evaluate exploitation of learned knowledge.
* Outcome: The agents perform well initially, exploiting the learned Q-values. However, they may get stuck in local optima and fail to explore new, potentially better, strategies.

**Exploit Run (8500 steps):**

* Objective: Evaluate balance between exploration and exploitation.
* Outcome: The agents perform similarly to the greedy run but with occasional exploration. This balance allows them to potentially discover better strategies while still exploiting learned knowledge.

## Experiment 2

### Experiment Summary:

This experiment aimed to evaluate the performance of three agents (Red, Blue, and Black) using the SARSA algorithm in a 5x5 grid environment. The agents were trained to navigate the environment to reach a goal state while avoiding obstacles. The Q-learning algorithm was used to update the Q-values in each agent's Q-table based on their interactions with the environment.

Pseudocode:

1. Define environment parameters: rows, cols, num\_actions, learning\_rate, discount\_factor.
2. Initialize agents with Q-tables.
3. Run exploit policy for 9000 steps:

* For each step, each agent takes an action based on the exploit policy.
* If an agent reaches the goal or an obstacle, reset its position to the start.

1. Display results:

* Print the final environment state after the exploit run.
* Print the rewards obtained by each agent.
* Print the Q-tables of each agent.

1. Plot the rewards obtained by each agent over time.

Steps Explanation:

* The environment is a 5x5 grid where each cell represents a state. The agents can move up, down, left, right, northeast, or northwest.
* The Q-table for each agent is initialized with zeros for each state-action pair.
* During the exploit run, each agent selects actions based on the exploit policy, which balances exploration and exploitation.
* If an agent reaches the goal state (5,5) or collides with an obstacle (1,4), it is reset to the start state (3,3).
* The Q-values in each agent's Q-table are updated using the SARSA algorithm, which considers the current state, action, reward, next state, and next action.
* The final Q-tables show the learned Q-values for each state-action pair, indicating the expected future reward for taking an action in a given state.
* The rewards obtained by each agent indicate their performance in navigating the environment. Negative rewards suggest that the agents took suboptimal actions

**Results Demonstration:**

* The exploit run resulted in the final environment state [5, 5, 1, 3, 3, 5, 0, 0, 0, 1, 4, 0, 0, 5, 5].
* The Red Agent received a reward of -2944, the Blue Agent received a reward of -2972, and the Black Agent received a reward of -2944.
* The Q-tables show that the agents have learned different strategies to navigate the environment, but there is room for improvement in their coordination and efficiency.

### Quality of Agent Coordination and Results:

* The Q-tables of the agents demonstrate learning and convergence towards better policies, especially in states where rewards are higher.
* Red and black agents' Q-values indicate learning of the optimal path towards the goal state (4,4), while blue agent's Q-values seem less optimal.
* The agents seem to coordinate reasonably well in the exploit run, as they collectively avoid invalid moves and move towards the goal.
* Further training and tuning of the SARSA algorithm parameters may lead to better coordination among the agents, resulting in improved performance.

## Experiment 3

### Experiment Summary:

This experiment aimed to investigate the impact of different learning rates (0.15 and 0.45) on the performance of three agents (Red, Blue, and Black) using the Q-learning algorithm with a pseudo-exploit policy. The agents were trained in a 5x5 grid environment to navigate towards a goal state while avoiding obstacles.

# Pseudocode:

1. Initialize environment parameters and agents with different learning rates.
2. Run exploit policy with learning rate 0.15 for 8500 steps:

* Each agent takes actions based on the pseudo-exploit policy.
* If an agent reaches the goal or an obstacle, reset its position.

1. Display results for learning rate 0.15:

* Final environment state and rewards obtained by each agent.
* Q-tables for each agent with learning rate 0.15.

1. Plot rewards obtained by each agent over time for learning rate 0.15.
2. Repeat steps 1-4 for learning rate 0.45.

Steps Explanation:

* The experiment compares the performance of agents trained with different learning rates (0.15 and 0.45) using the Q-learning algorithm with a pseudo-exploit policy.
* Each agent learns a Q-table representing the expected future rewards for each action in a given state.
* During the exploit policy run, agents select actions based on the pseudo-exploit policy, balancing exploration and exploitation to navigate towards the goal state.
* The learning rate affects how much the agent updates its Q-values after each action, influencing the speed and stability of learning.

**Results Demonstration:**

### **Learning Rate 0.15:**

* The exploit run resulted in a final environment state [1, 3, 1, 4, 2, 5, 0, 1, 0, 5, 0, 5, 0, 0, 4].
* Red Agent: Reward -2777, Blue Agent: Reward -2707, Black Agent: Reward -2721.
* The Q-tables show learned Q-values for each agent-state-action pair, indicating their learned strategies.
* The rewards suggest that the agents achieved moderate performance with learning rate 0.15.

### **Learning Rate 0.45:**

* The exploit run resulted in a final environment state [2, 4, 1, 5, 1, 3, 1, 0, 0, 5, 1, 4, 0, 0, 4].
* Red Agent: Reward -2791, Blue Agent: Reward -2749 Black Agent: Reward -2665.
* The Q-tables show learned Q-values for each agent-state-action pair, indicating their learned strategies.
* The reward plots show that agents with a learning rate of 0.45 achieve higher cumulative rewards over the exploit run compared to those with a learning rate of 0.15.

### Insights of Learning Rates

The experiment demonstrates the impact of learning rate on the convergence and performance of agents using Q-learning with a policy exploitation strategy. A higher learning rate allows agents to update their Q-values more aggressively, potentially leading to faster convergence and better performance. However, choosing an optimal learning rate depends on the specific characteristics of the environment and the task at hand, as overly high learning rates may result in instability or oscillations in learning. In this experiment, a learning rate of 0.45 outperformed a learning rate of 0.15 in terms of cumulative rewards, indicating the importance of selecting an appropriate learning rate for achieving efficient learning in reinforcement learning tasks.

## Experiment 4

### Experiment Summary:

Experiment 4 aimed to train three agents (Red, Blue, and Black) using Q-learning and ε-greedy exploration in a 5x5 grid world environment. The agents were trained to navigate the environment and maximize their rewards by learning optimal policies. The experiment consisted of two phases: random exploration and exploitation. In the random exploration phase, the agents took random actions to gather initial experience, while in the exploitation phase, they exploited their learned knowledge to maximize rewards.

### Pseudocode:

1. Initialize the environment, agents, and hyperparameters.
2. Perform 500 random exploration steps using a policy of "PRANDOM".
3. Perform 8500 exploitation steps using a policy of "PEXPLOIT" for each agent.
4. Evaluate the performance of the agents by calculating their rewards.
5. Display the Q-tables for each agent.
6. Plot the rewards obtained by each agent during the experiment.

Steps Explanation:

* Initialization: Initialize the environment, agents, and hyperparameters required for the experiment.
* Random Exploration: Allow the agents to explore the environment randomly for 500 steps to gather initial experience.
* Exploitation: Use ε-greedy exploitation to allow the agents to learn optimal policies by exploiting their current knowledge for 8500 steps.
* Evaluation: Evaluate the performance of the agents by calculating their rewards obtained during the exploitation phase.
* Q-table Display: Display the Q-tables for each agent to visualize their learned policies.
* Reward Plotting: Plot the rewards obtained by each agent over the course of the experiment to analyze their learning progress.

**Results Demonstration:**

The experiment results show that the agents were able to learn effective policies for navigating the environment, as evidenced by their accumulated rewards. The Q-tables provide insight into the agents' learned policies for each state-action pair, indicating their understanding of the environment dynamics

### Insights and Results

**Performance Comparison:**

* The agents' performance was evaluated based on their rewards accumulated during the exploitation phase.
* Red Agent: -5552, Blue Agent: -5440, Black Agent: -5426.
* Red Agent performed slightly worse than the other two agents, possibly due to its exploration strategy or initial Q-table setup.
* Blue and Black Agents performed similarly, indicating that their strategies were more effective in this environment.

**Learned Policies:**

* Q-tables provide insights into the agents' learned policies for each state-action pair.
* Agents learned to avoid actions with negative rewards and prioritize actions that lead to positive rewards.
* The Q-tables show that agents learned different strategies for navigating the environment, leading to varying levels of performance.

**Exploration vs. Exploitation:**

* The experiment demonstrated the balance between exploration and exploitation in reinforcement learning.
* Random exploration helped agents gather initial experience, while exploitation allowed them to refine their policies based on learned knowledge.

**Agent Coordination:**

* The experiment did not explicitly focus on agent coordination, but the agents' individual performances can indicate their ability to coordinate actions to achieve a common goal.
* The agents' rewards suggest that they were able to coordinate their actions to some extent, as they achieved similar performance levels.

**Future Improvements:**

* To improve the agents' performance, one could experiment with different hyperparameters, such as learning rate and discount factor, or explore other exploration strategies.
* Fine-tuning the initial Q-table setup and exploration-exploitation trade-off could also lead to better performance.

**Overall Impression:**

* The experiment demonstrates the effectiveness of Q-learning and ε-greedy exploration in training agents to perform well in a grid world environment.
* While there is room for improvement, the agents' ability to learn effective policies and maximize rewards highlights the potential of these techniques in reinforcement learning tasks.

### References

1. <https://github.com/rhodesnoahd/ai-reinforcement-learning/blob/master/action.py>
2. <https://towardsdatascience.com/reinforcement-learning-teach-a-taxi-cab-to-drive-around-with-q-learning-9913e611028f>
3. <https://github.com/mbraihan/path-planning-using-reinforcement-learning>
4. <https://github.com/sichkar-valentyn/Reinforcement_Learning_in_Python/tree/master>
5. <https://github.com/schneblizoltan/multiagent_rf>