RL Part2

Jai advith

February 2025

1 Deterministic

1.1 Q Learning

1.1.1 Applied Q Learning to the deterministic environment using the following parameters

- $\alpha = 0.1$: Learning rate
- $\gamma = 0.99$: Discount factor
- $\epsilon = 1.0$: Initial exploration rate
- $epsilon\_decay = 0.995$: Rate at which exploration decreases
- $\min\_epsilon = 0.01$: Minimum exploration rate
- $num\_episodes = 10$: Number of training episodes

For each episode:

1. Reset environment
2. Loop until episode is done:
   (a) Select action using epsilon-greedy policy
   (b) Take action and observe next state and reward
   (c) Update Q-value using Q-learning update rule:
   (d) $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)]$
   (e) Move to next state
3. Decay epsilon for less exploration over time
4. Track rewards and epsilon values

1.1.2 Performing it gave: Episode 100/100, Avg Reward: 423.80, Epsilon: 0.606

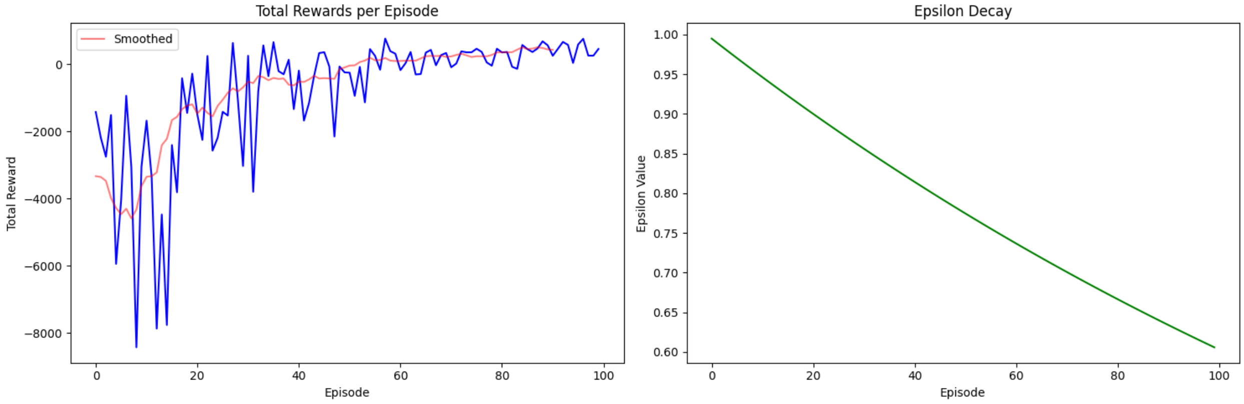


Figure 1: Plot of Total Rewards per episode and Epsilon Decay

1.2 Run environment for 10 episodes using greedy policy

We got a consistent graph, each episode had a total reward of 787.

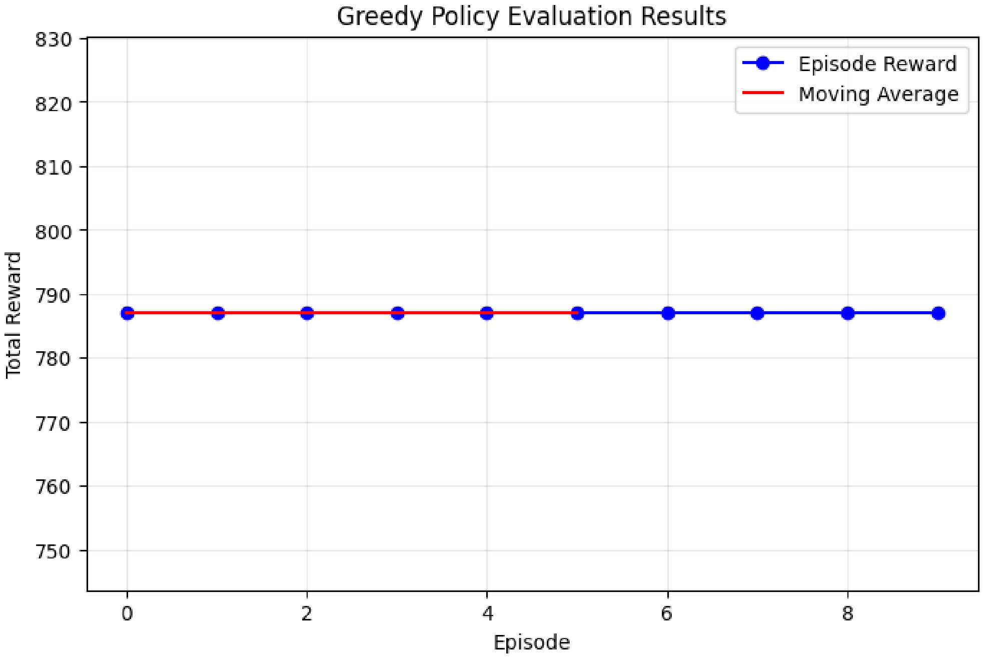
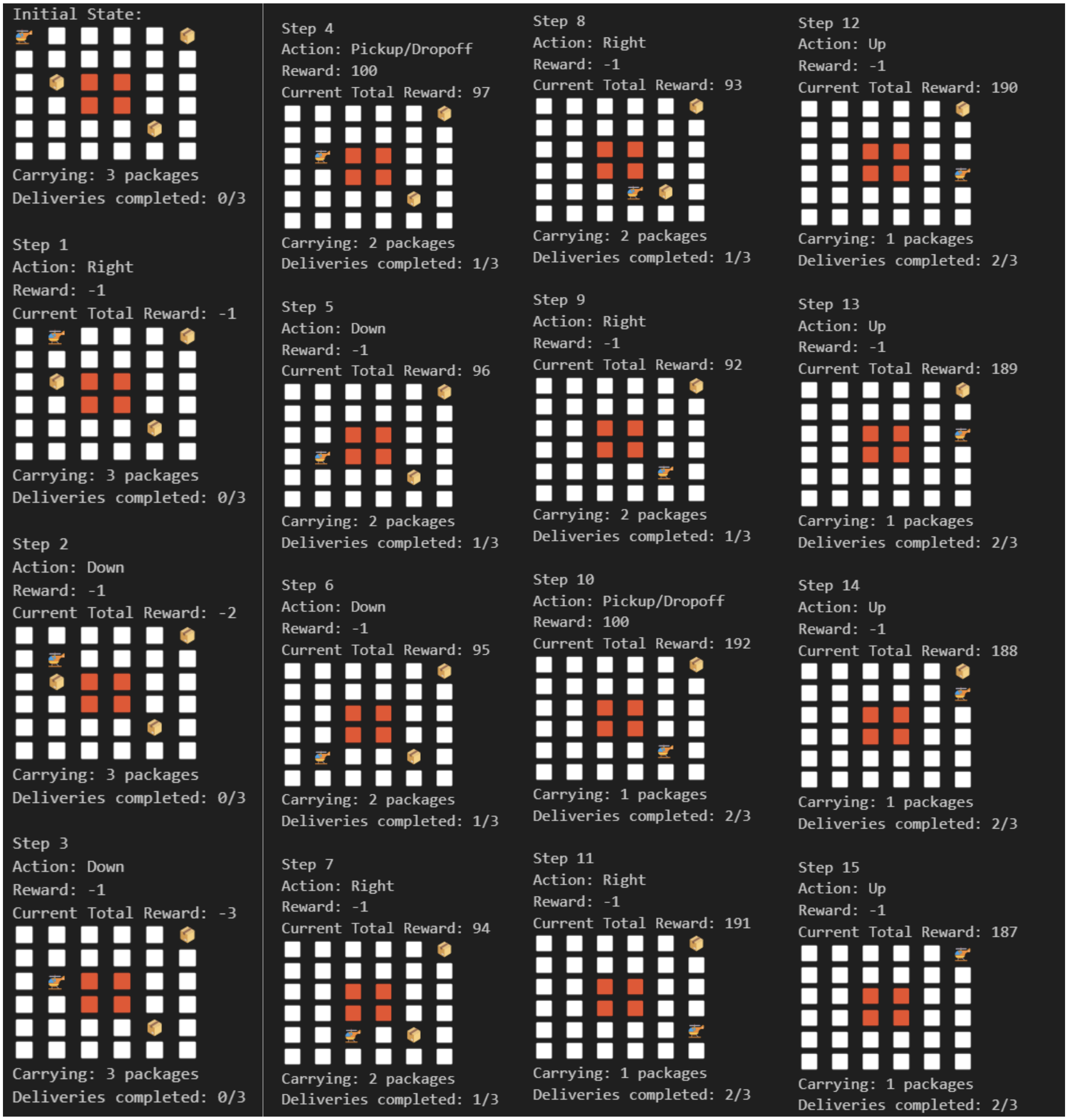


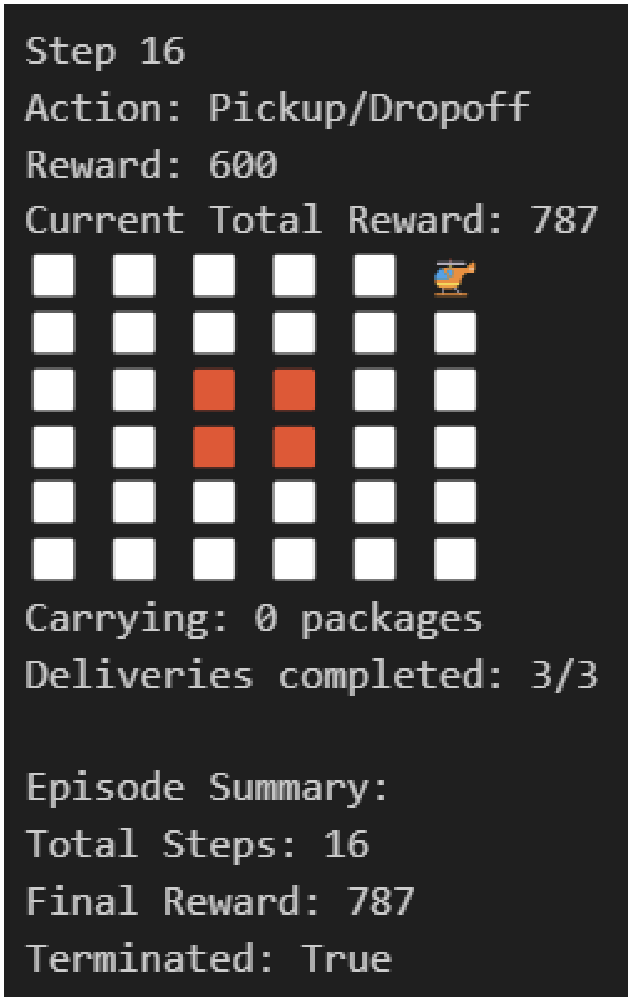
Figure 2: Plot of Total Rewards per episode in greedy policy

1.3 Run environment for 1 episode and render each step



```
Initial State:
🚁 ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ 📦 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 3 packages
Deliveries completed: 0/3

Step 1
Action: Right
Reward: -1
Current Total Reward: -1
⬜ 🚁 ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ 📦 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 3 packages
Deliveries completed: 0/3

Step 2
Action: Down
Reward: -1
Current Total Reward: -2
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ 🚁 ⬜ ⬜ ⬜ ⬜
⬜ 📦 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 3 packages
Deliveries completed: 0/3

Step 3
Action: Down
Reward: -1
Current Total Reward: -3
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ 🚁 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 3 packages
Deliveries completed: 0/3

Step 4
Action: Pickup/Dropoff
Reward: 100
Current Total Reward: 97
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ 🚁 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 5
Action: Down
Reward: -1
Current Total Reward: 96
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ 🚁 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 6
Action: Down
Reward: -1
Current Total Reward: 95
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ 🚁 ⬜ ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 7
Action: Right
Reward: -1
Current Total Reward: 94
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🚁 ⬜ 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 8
Action: Right
Reward: -1
Current Total Reward: 93
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ 🚁 📦 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 9
Action: Right
Reward: -1
Current Total Reward: 92
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 🚁 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 2 packages
Deliveries completed: 1/3

Step 10
Action: Pickup/Dropoff
Reward: 100
Current Total Reward: 192
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ 🚁 ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3

Step 11
Action: Right
Reward: -1
Current Total Reward: 191
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ 🚁
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3

Step 12
Action: Up
Reward: -1
Current Total Reward: 190
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ 🚁
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3

Step 13
Action: Up
Reward: -1
Current Total Reward: 189
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ 🚁
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3

Step 14
Action: Up
Reward: -1
Current Total Reward: 188
⬜ ⬜ ⬜ ⬜ ⬜ 📦
⬜ ⬜ ⬜ ⬜ ⬜ 🚁
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3

Step 15
Action: Up
Reward: -1
Current Total Reward: 187
⬜ ⬜ ⬜ ⬜ ⬜ 🚁
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ 🟥 🟥 ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
⬜ ⬜ ⬜ ⬜ ⬜ ⬜
Carrying: 1 packages
Deliveries completed: 2/3
```

Figure 3: steps



```
Step 16
Action: Pickup/Dropoff
Reward: 600
Current Total Reward: 787
⬜⬜⬜⬜⬜🚁
⬜⬜⬜⬜⬜⬜
⬜⬜🟥🟥⬜⬜
⬜⬜🟥🟥⬜⬜
⬜⬜⬜⬜⬜⬜
⬜⬜⬜⬜⬜⬜
Carrying: 0 packages
Deliveries completed: 3/3

Episode Summary:
Total Steps: 16
Final Reward: 787
Terminated: True
```

Figure 4: Final step

1.4 SARSA

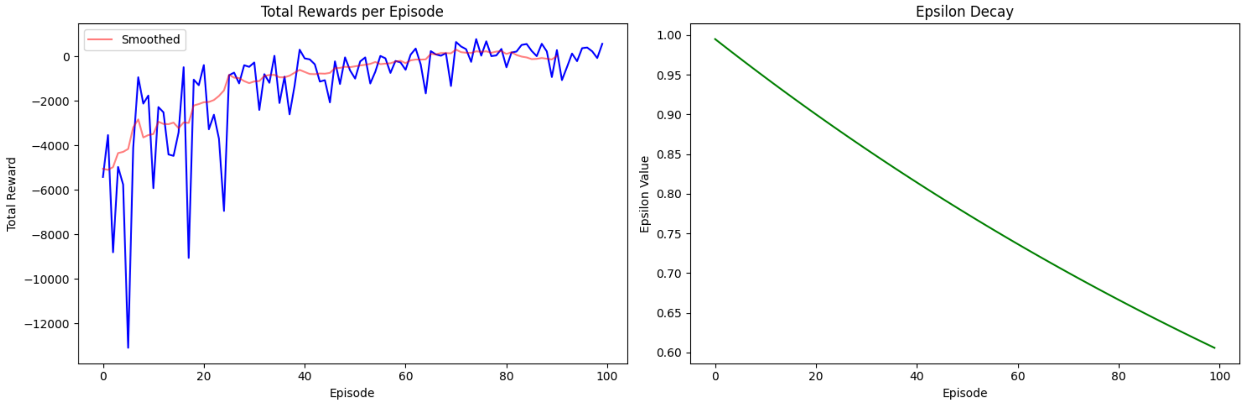


Figure 5: SARSA plots

1.5 Hyperparameter tuning

1.5.1 Gamma

Results Summary:

Gamma = 0.9 Mean Reward: 373.90 Std Reward: 184.94
Gamma = 0.95 Mean Reward: 352.00 Std Reward: 229.59
Gamma = 0.99 Mean Reward: 384.80 Std Reward: 167.84
Best gamma value: 0.99

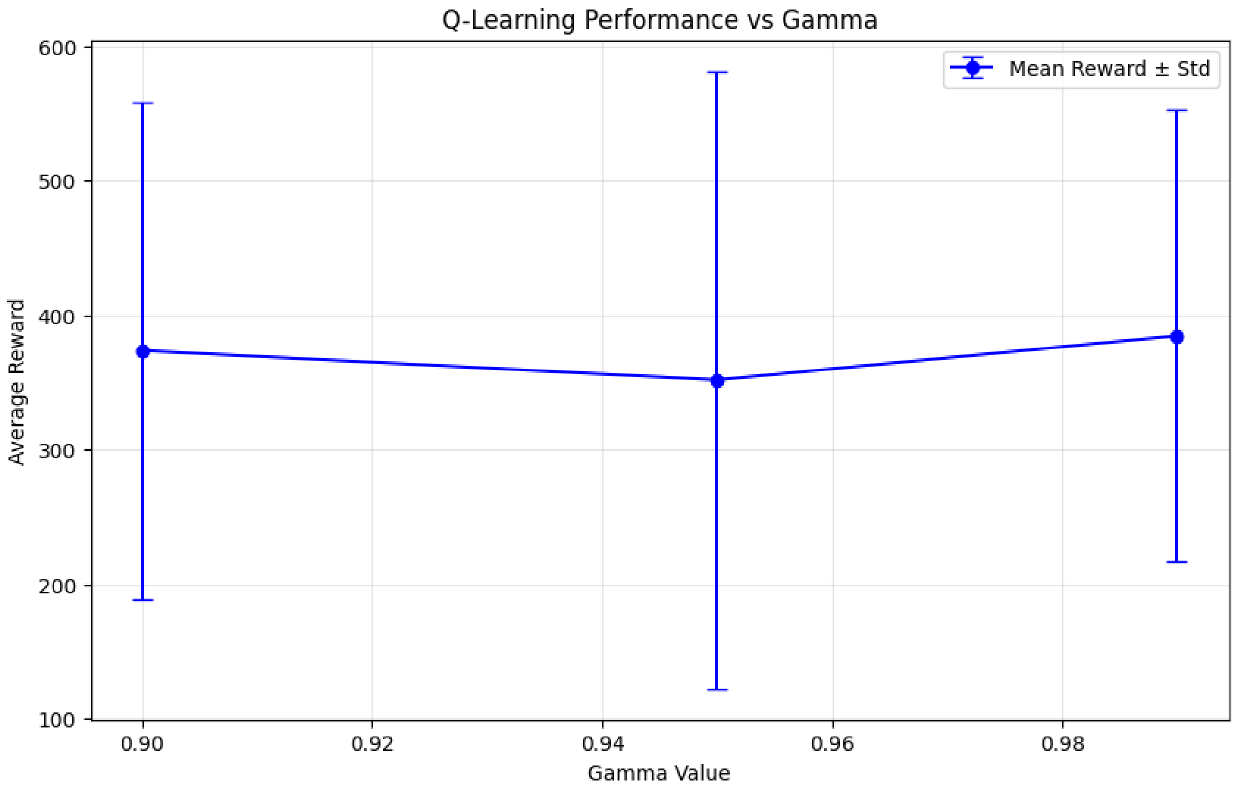


Figure 6: Gammma results

1.5.2 Epsilon Decay

Results Summary:
Epsilon Decay = 0.99 Mean Reward: 613.90 Std Reward: 130.59
Epsilon Decay = 0.995 Mean Reward: 316.10 Std Reward: 181.41
Epsilon Decay = 0.999 Mean Reward: -2563.90 Std Reward: 1562.55
Best epsilon decay value: 0.99

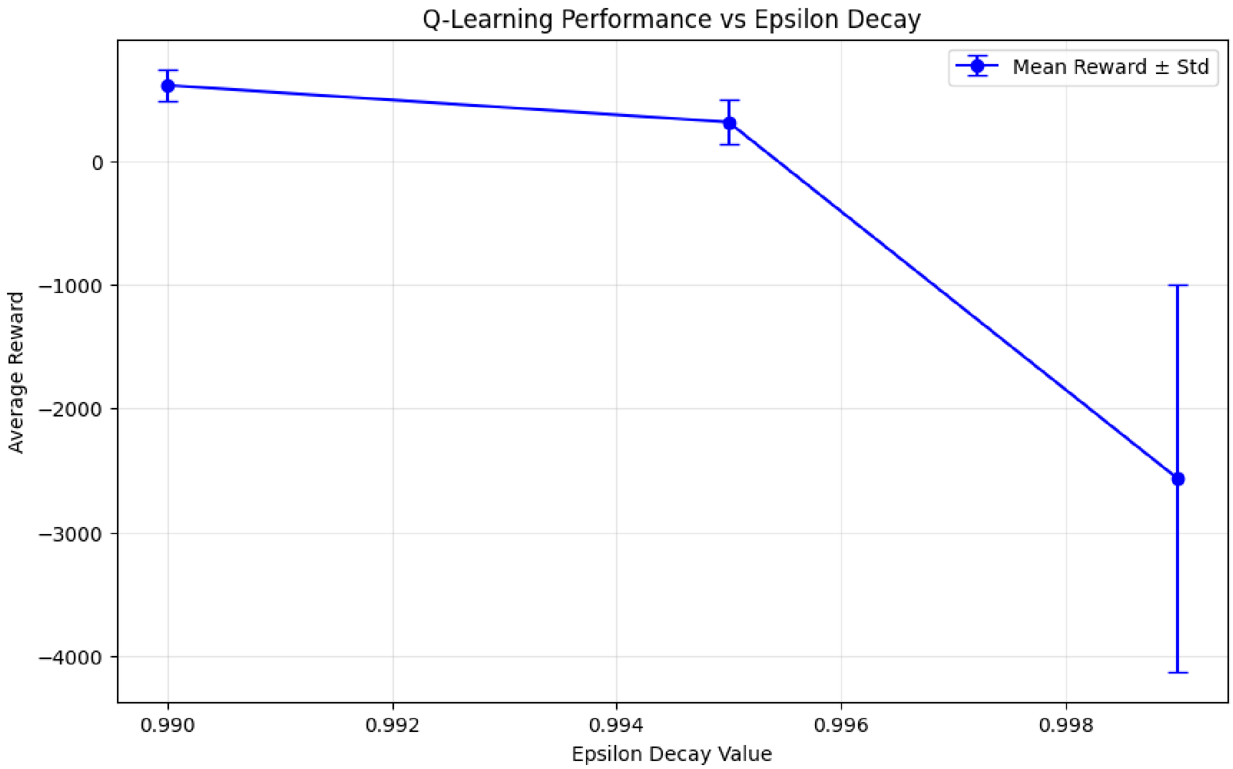


Figure 7: results

2 Stochastic

2.1 Hyperparameter tuning

2.1.1 Gamma

Results Summary:
Gamma = 0.9 Mean Reward: 298.90 Std Reward: 358.94
Gamma = 0.95 Mean Reward: 456.80 Std Reward: 161.77
Gamma = 0.99 Mean Reward: 220.20 Std Reward: 449.49
Best gamma value: 0.95

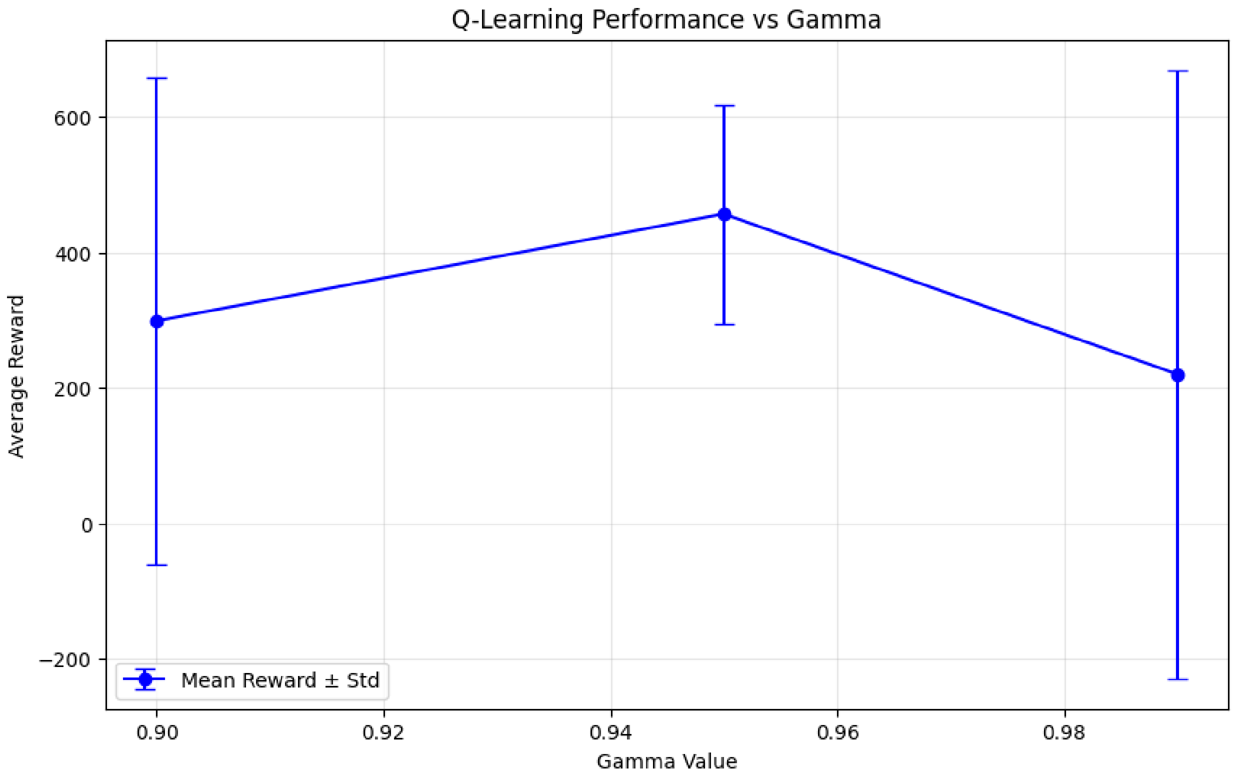


Figure 8: Gammma results

2.1.2 Epsilon Decay

Results Summary:

Epsilon Decay = 0.99 Mean Reward: 694.00 Std Reward: 78.57
Epsilon Decay = 0.995 Mean Reward: 314.50 Std Reward: 316.72
Epsilon Decay = 0.999 Mean Reward: -1703.10 Std Reward: 1194.23
Best epsilon decay value: 0.99

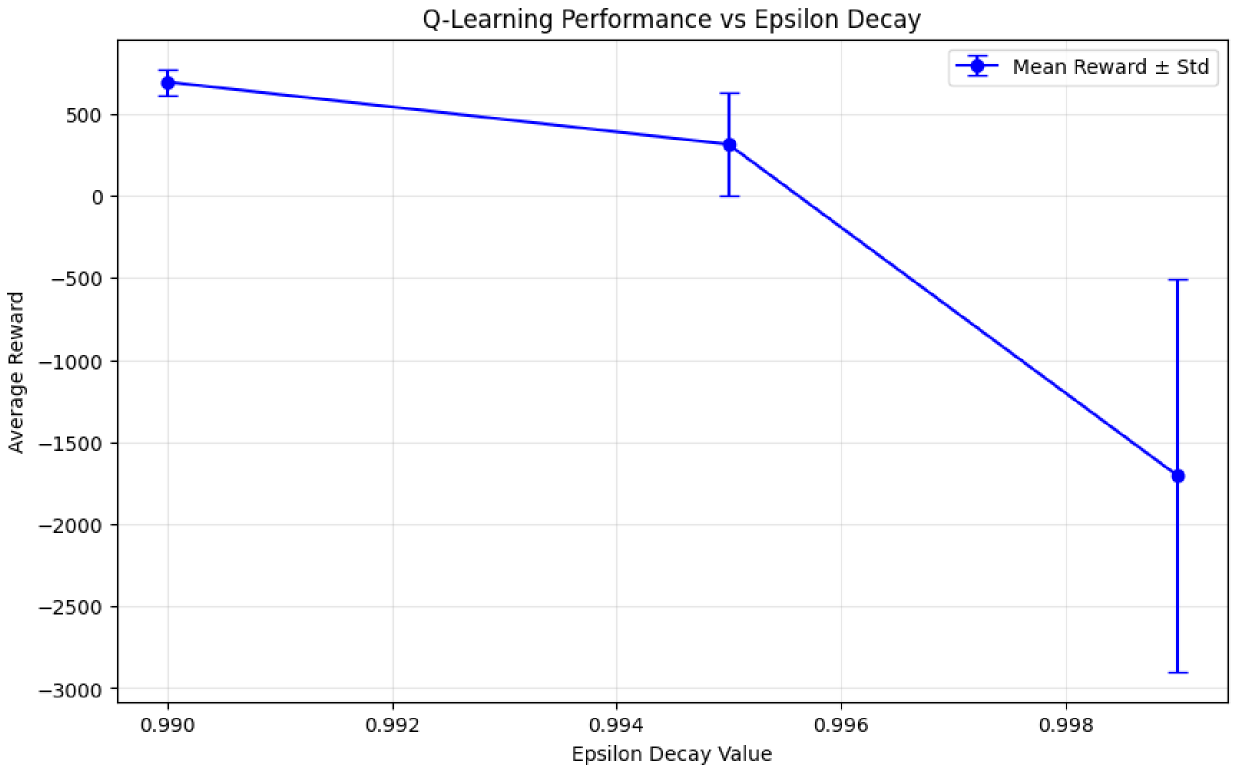


Figure 9: results

2.2 Q Learning

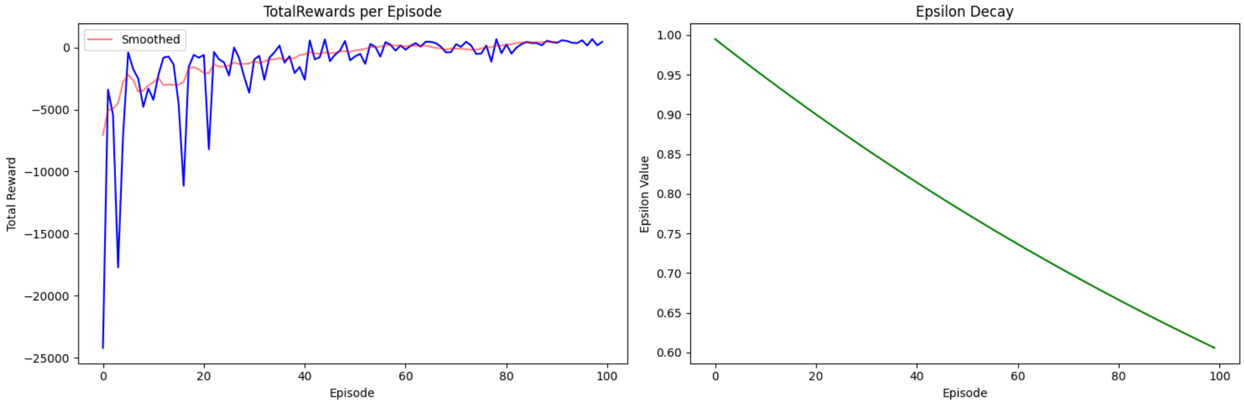


Figure 10: Plot of Total Rewards per episode and Epsilon Decay

2.3 SARSA

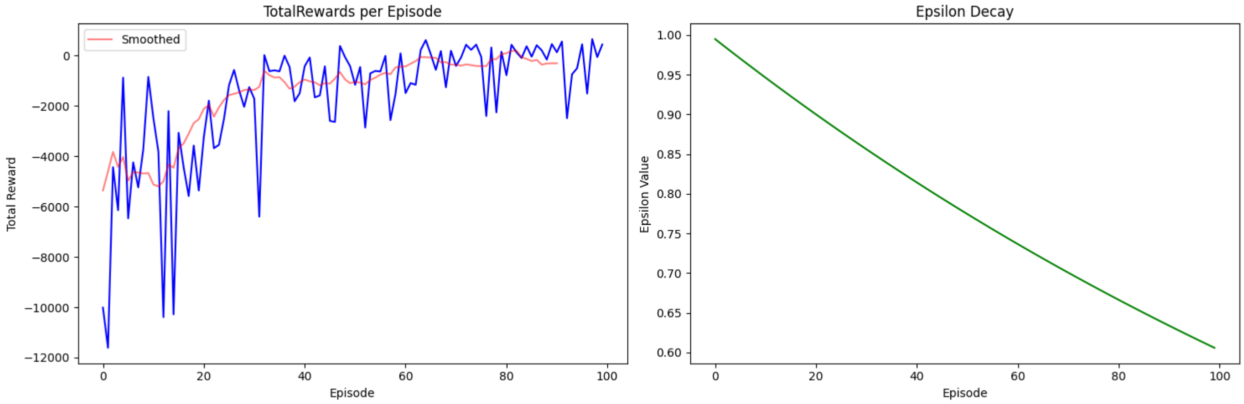


Figure 11: Plot of Total Rewards per episode and Epsilon Decay

3 Comparing on Deterministic env

Final Results: Q-Learning - Mean reward (last 10 episodes): 220.20 SARSA - Mean reward (last 10 episodes): 373.90

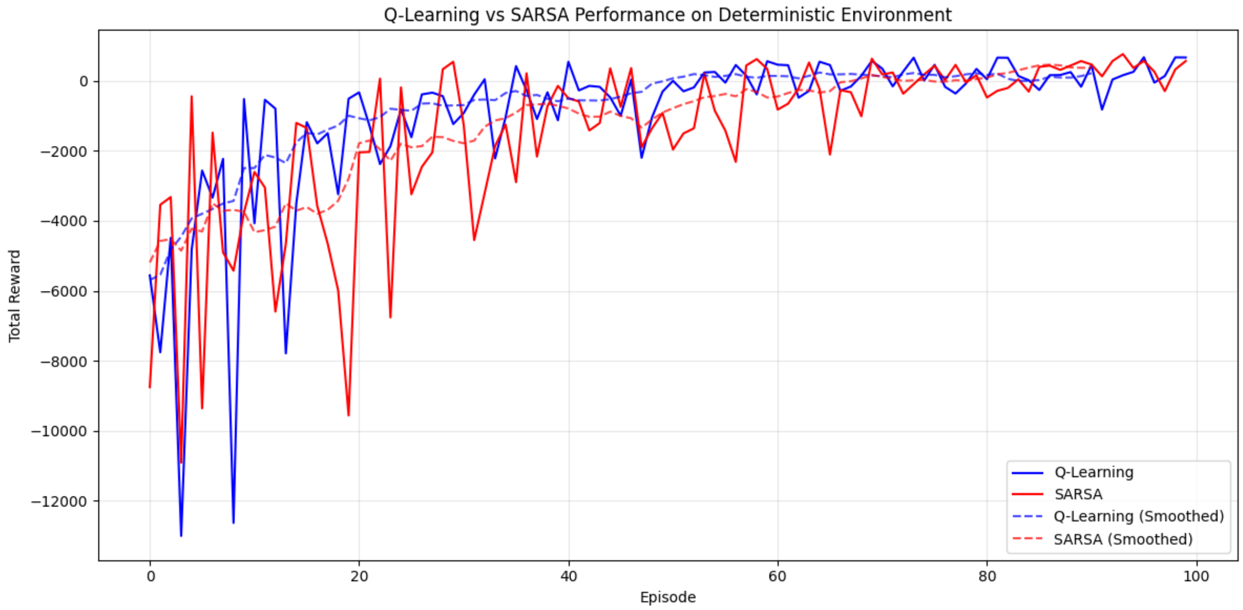


Figure 12: Q Learning vs SARSA

Interpretation of Results:

In this deterministic environment:

1. Learning Dynamics: The graph shows how Q-Learning and SARSA learn over time. Both algorithms eventually learn effective policies, but their learning trajectories differ.

2. Off-policy vs On-policy: Q-Learning, being off-policy, learns the optimal policy regardless of the

exploration policy being used. SARSA, being on-policy, learns a policy that considers the exploration strategy.

3. Performance: In a fully deterministic environment, Q-Learning tends to converge to the optimal policy faster as it directly learns from the best next action. SARSA is more conservative as it considers the actual (potentially exploratory) next action.

4. Stability: SARSA typically shows more stable learning curves because it accounts for the exploration strategy in its updates, rather than assuming optimal future choices.

5. Final Policy: Both algorithms should eventually converge to similar policies in a deterministic environment with sufficient training.

4 Comparing Q Learning vs SARSA on stochastic env

Final Results: Q-Learning - Mean reward (last 10 episodes): 238.00 SARSA - Mean reward (last 10 episodes): 70.60

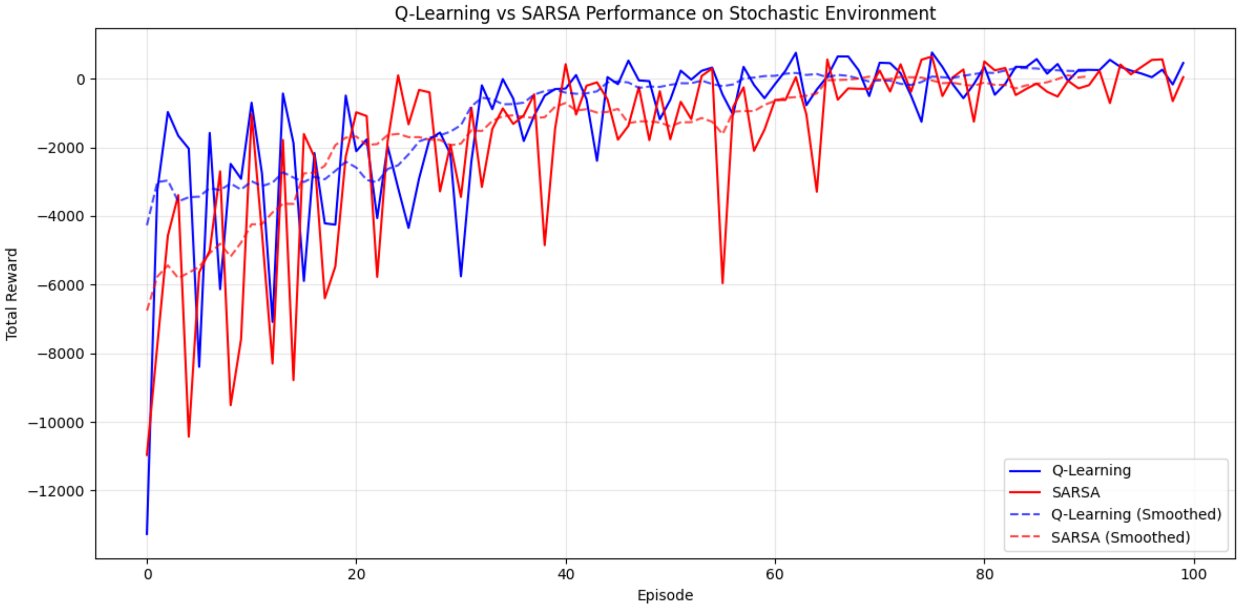


Figure 13: Q Learning vs SARSA

Interpretation of Results:

In this stochastic environment:

1. Learning Dynamics: The comparison between Q-Learning and SARSA reveals how each algorithm adapts to uncertainty in action outcomes.

2. Off-policy vs On-policy: SARSA's on-policy nature provides a significant advantage in stochastic environments, as it learns a policy that accounts for the randomness in action results.

3. Performance: Q-Learning might show more variability in performance since it always optimistically assumes the best next action will be taken, which isn't always possible in a stochastic environment.

4. Safety: SARSA tends to learn safer policies in stochastic environments, as it directly incorporates the uncertainty of exploration into its value updates.

5. Long-term benefits: While Q-Learning might appear to learn faster initially, SARSA often produces more reliable policies in stochastic domains where the optimistic assumptions of Q-Learning can lead to risky behavior.

5 Explanation of Tabular Methods

5.1 Q-Learning

Q-Learning is an off-policy temporal difference (TD) learning algorithm that approximates the optimal action-value function $Q^*(s, a)$ regardless of the agent's policy.

5.1.1 Update function

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s,a)\right]$$

5.1.2 Key features

- **Off-policy**: Learns the optimal policy while following an exploration policy.
- Uses maximum estimated value for the next state ($\max_{a'} Q(s', a')$).
- Tends to be more aggressive/optimistic in its value estimates.
- In our implementation, we used epsilon-greedy exploration with decay.

5.2 SARSA

SARSA is an on-policy TD learning algorithm that learns the value of the policy being followed.

5.2.1 Update function

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma Q(s', a') - Q(s,a)\right]$$

5.2.2 Key features

- **On-policy**: Learns values for the policy it's actually following.
- Uses the actual next action's value for updates.
- Generally more conservative than Q-learning.
- Better suited for stochastic environments.

5.3 Key Components of Both Implementations

- State encoding: Reduced state space by using position and delivery status only.
- Epsilon-greedy exploration strategy with decaying epsilon.
- Hyperparameter tuning for alpha (learning rate) and gamma (discount factor).

6 Criteria for Good Reward Function

A good reward function should be:

1. **Sparse but informative**: In our environment, we used -1 for each step (encouraging efficiency), -100 for no-fly zones (strongly discouraging dangerous actions), +100 for successful deliveries, and +500 for completing all tasks (encouraging goal completion).

2. **Aligned with task objectives**: The rewards directly correspond to the goal of delivering packages efficiently while avoiding restricted areas.

3. **Properly scaled**: The relative magnitudes of rewards (-1 vs -100 vs +100 vs +500) reflect their importance in the task hierarchy.

4. **Consistent between environments**: Both deterministic and stochastic environments use the same reward structure, allowing for fair comparison of algorithms.

The reward function we used successfully shaped the behavior of both algorithms to complete deliveries while avoiding no-fly zones. In the stochastic environment, we saw that SARSA tended to learn more robust policies due to its on-policy nature, better accounting for the uncertainty of actions.