Indian Institute of Technology, Madras
Chennai, Tamil Nadu

CS6700: Reinforcement Learning, Report on programming assignment 1

Jan-May 2024

Submitted by: Nataraj Das (CS23D404)
&
Shuvrajeet Das (CS23E001)

0.1 Why is the start state important in a greed world RL problem? [3]

- Initial Exploration: The choice of start state influences where the agent begins its exploration of the environment. Different start states may lead the agent to encounter different parts of the environment first, which can affect the trajectory of learning. For example, starting in a state with high rewards may lead to faster initial learning, while starting in a state with low rewards may require more exploration before discovering.
- Learning Dynamics: The initial state affects the sequence of state-action pairs encountered during learning. Since SARSA is an on-policy algorithm, the agent's policy during learning is directly influenced by the state-action pairs it encounters. Changing the start state can lead to different sequences of experiences, which in turn can affect how quickly and effectively the agent learns.
- Generalization: The agent's learned policy may generalize differently depending on the start state. If the agent starts in a state that is representative of the entire environment, it may learn a more general policy that applies well across different starting conditions. Conversely, starting in a specific state may lead to a policy that is tailored to that particular starting condition and may not generalize as effectively to other scenarios.
- Convergence: The choice of start state can impact the convergence of the learning process. In some cases, certain start states may lead to faster convergence to an optimal policy, while others may result in slower convergence or convergence to sub-optimal policies.

Overall, changing the start state can influence the exploration-exploitation trade-off, the learning trajectory, and the final learned policy of the agent. It's an important consideration in reinforcement learning experiments and may require experimentation to find the most appropriate starting conditions for a given problem. optimal policies.

0.2 Effects of stochasticity for the start state [3]

- Deterministic start state: In a deterministic grid world with no wind, the agent starts from a fixed initial state every episode. This provides consistency in the learning process as the agent begins from the same state, allowing it to focus on learning the optimal policy from that specific starting point. Since the starting state is deterministic, the agent's experiences are more predictable, and it can efficiently learn the consequences of its actions in that environment.
- Stochastic start state: When the start state is stochastic due to wind or other factors, the agent may begin from different initial states across episodes. This introduces variability in the learning process. Stochasticity in the start state can lead to the exploration of different parts of the environment, which can be beneficial for discovering diverse state-action pairs and understanding the dynamics of the environment better. However, it can also pose challenges for the agent, as it needs to adapt to different starting conditions. The agent may need to generalize its learned policy across various starting states, which can increase the complexity of the learning task.

Precisely, stochasticity in the start state introduces exploration and variability into the learning process, which can both enrich the agent's experience and present challenges in learning a robust policy. RL algorithms need to account for such variability and adapt accordingly to achieve optimal performance in stochastic environments.

0.3 Grid world setting

As per the question, the following experiments have been set over the environment with the following setting:

- 4 deterministic actions("up", "down","left", "right")
- consists of wind blowing that can push the agent one additional cell to the right after transitioning to the new state with a probability of 0.4.
- An episode is terminated either when a goal is reached or when the timesteps exceed 100.
- The dimensions of the grid are 10×10. The following types of states exist:
  1. Start state: The agent starts from this state.
  2. Goal state: The goal is to reach one of these states. There are 3 goal states in total.
  3. Obstructed state: These are walls that prevent entry to the respective cells. Transition to these states will not result in any change.
  4. Bad state: Entry into these states will incur a higher penalty than a normal state.
  5. Restart state: Entry into these states will incur a very high penalty and will cause the agent to teleport to the start state without the episode ending. Once the restart state is reached, no matter what action is chosen, it goes to the start state at the next step.
  6. Normal state: None of the above. Entry into these states will incur a small penalty.

0.4 Behavior of SARSA on policy algorithm [2]

[2]

Algorithm 1 SARSA: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:
Sates $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
$\lambda \in [0, 1]$: Trade-off between TD and MC
procedure SARSA($\mathcal{X}$, A, R, T, α, γ, λ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Select $(s, a) \in \mathcal{X} \times \mathcal{A}$ arbitrarily
    while s is not terminal **do**
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow T(s, a)$ ▷ Receive the new state
      Calculate π based on Q (e.g. epsilon-greedy)
      $a' \leftarrow \pi(s')$
      $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma Q(s', a'))$
      $s \leftarrow s'$
      $a \leftarrow a'$
    end while
  end whilereturn Q
end procedure

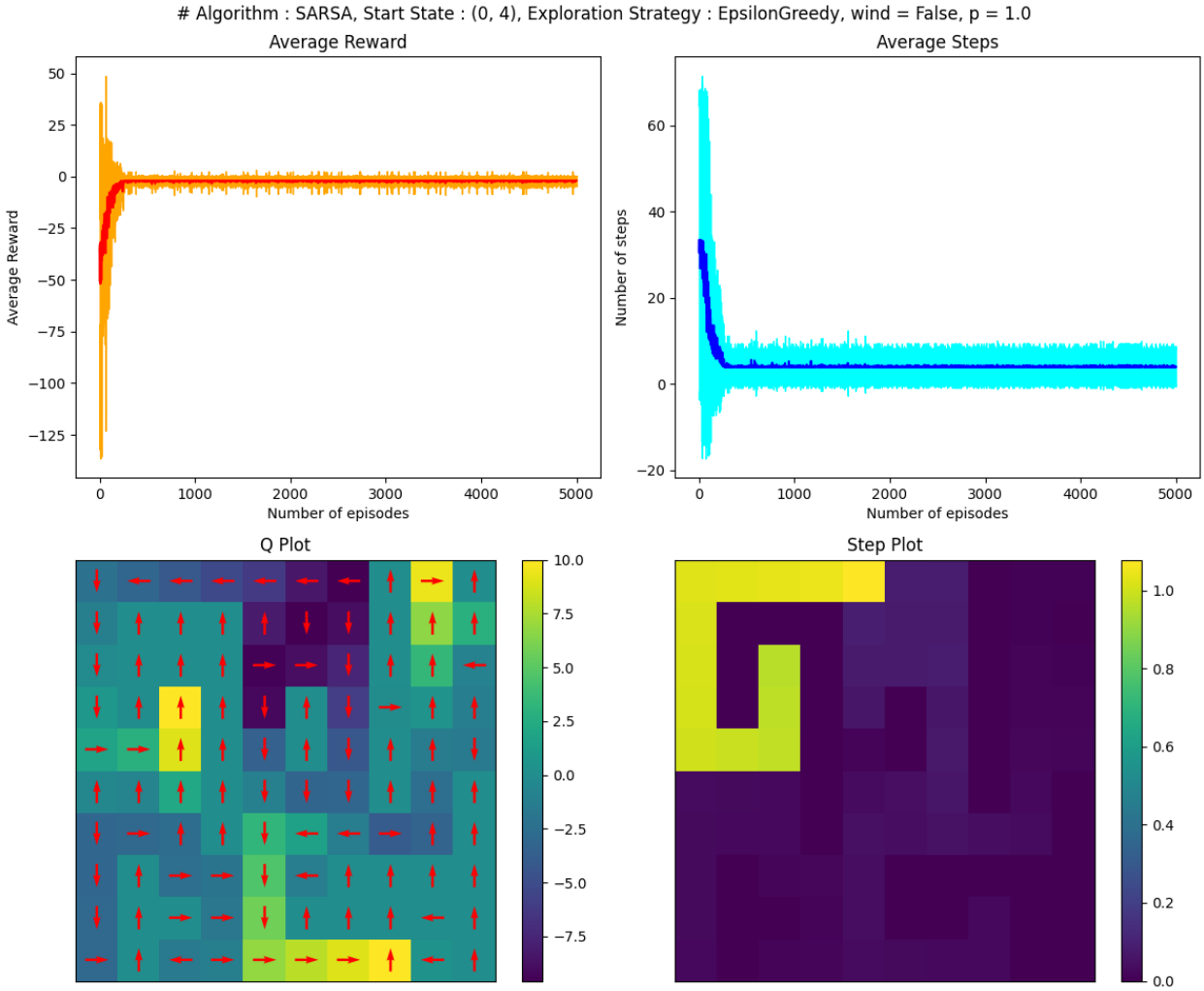

Figure 1: Sarsa behavior, start_state=(0,4), wind=False(p1.0)

From the state-visit count plot it is clear that the agent visits the upper left corner frequently. Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

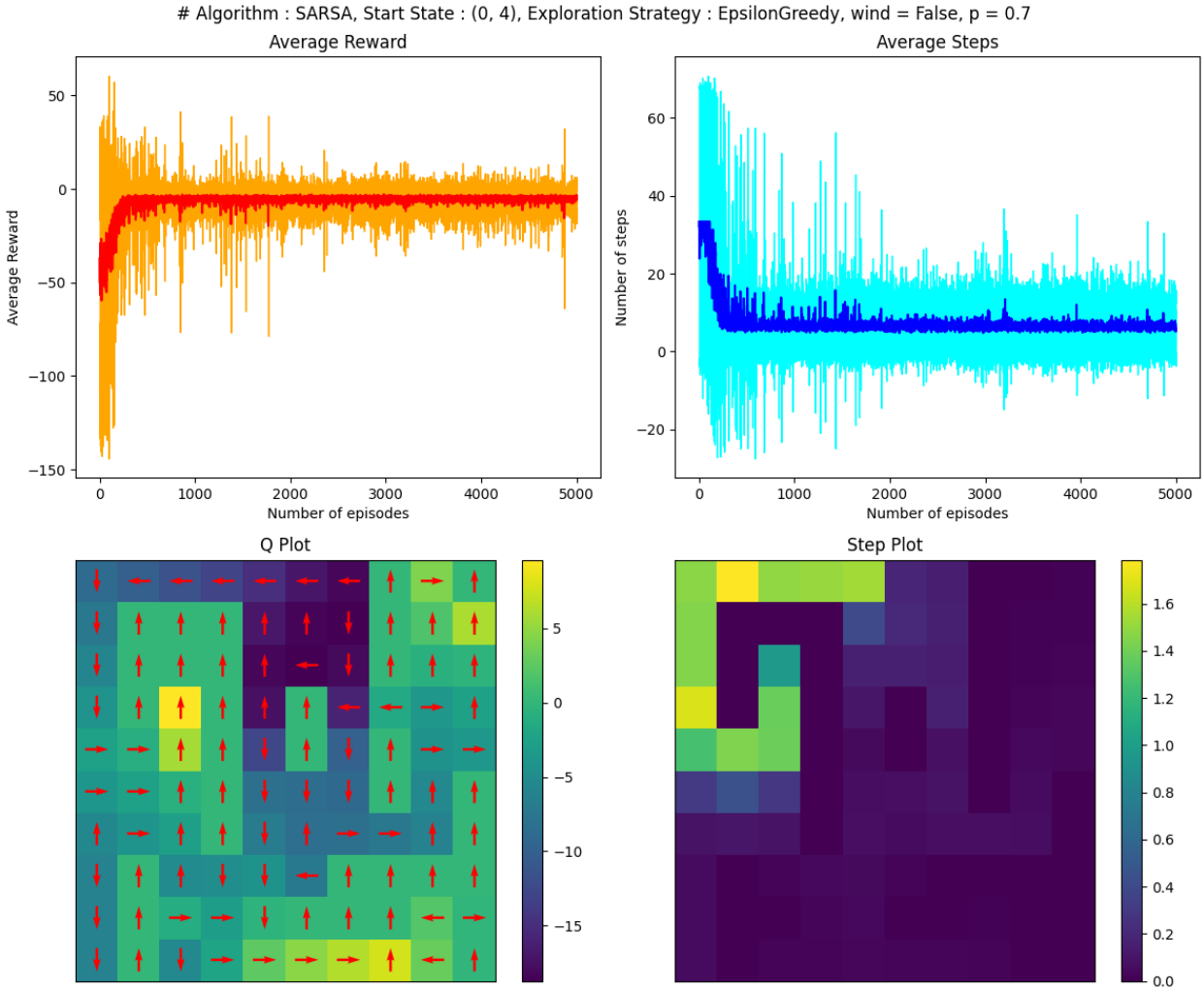

Figure 2: Sarsa behavior, start_state=(0,4), wind=False(p$\bar{0}$.7)

From the state-visit count heat-map it is clear that the agent visits the upper left corner very often. It has learnt to reach the top left corner goal state. It rarely visits the other two goal states.

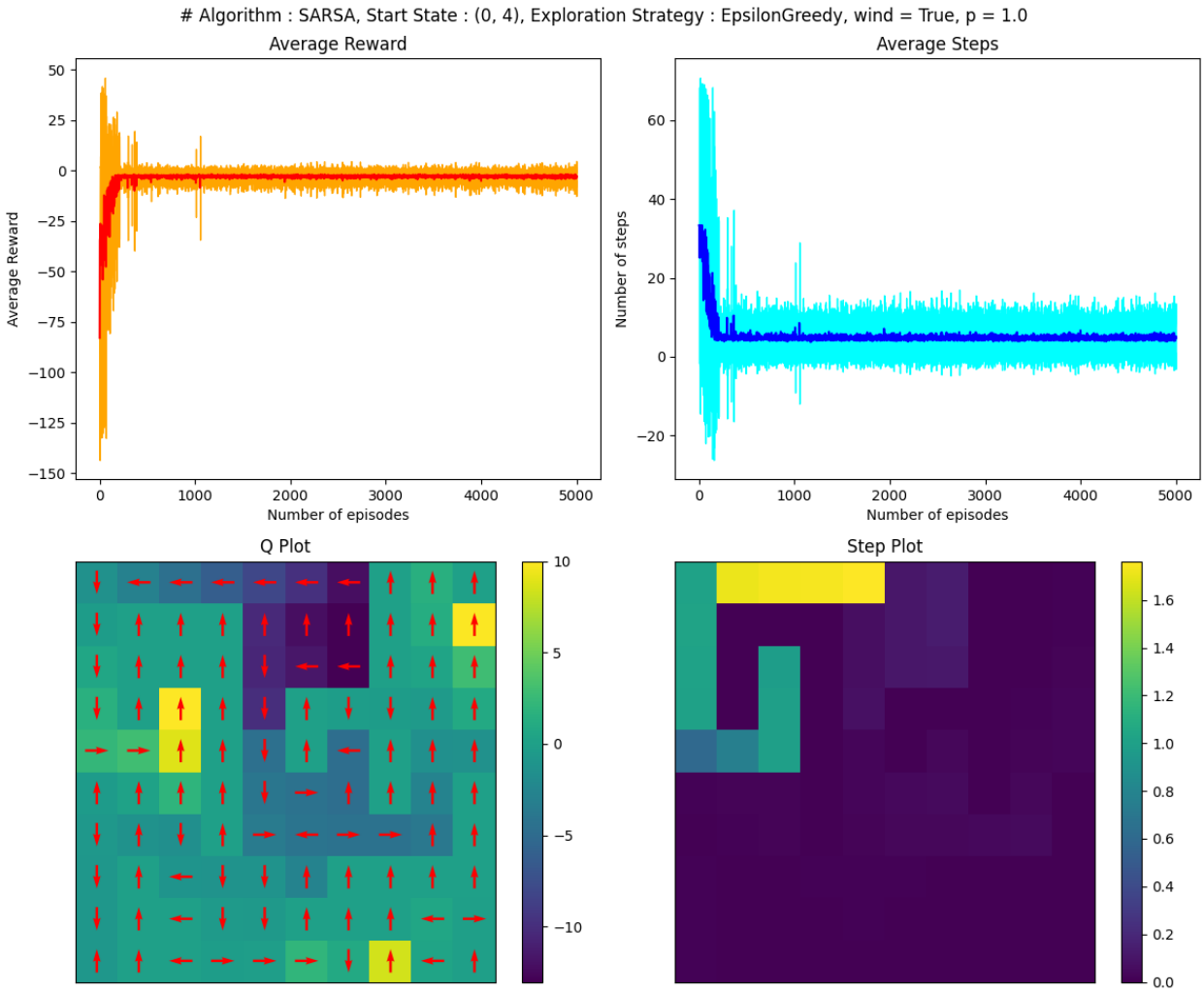

Figure 3: Sarsa behavior, start_state=(0,4), wind=True(p1.0)

From the state-visit count heatmap it is clear that the agent visits the top left corner very frequently. It has learnt to reach the upper left corner goal state.

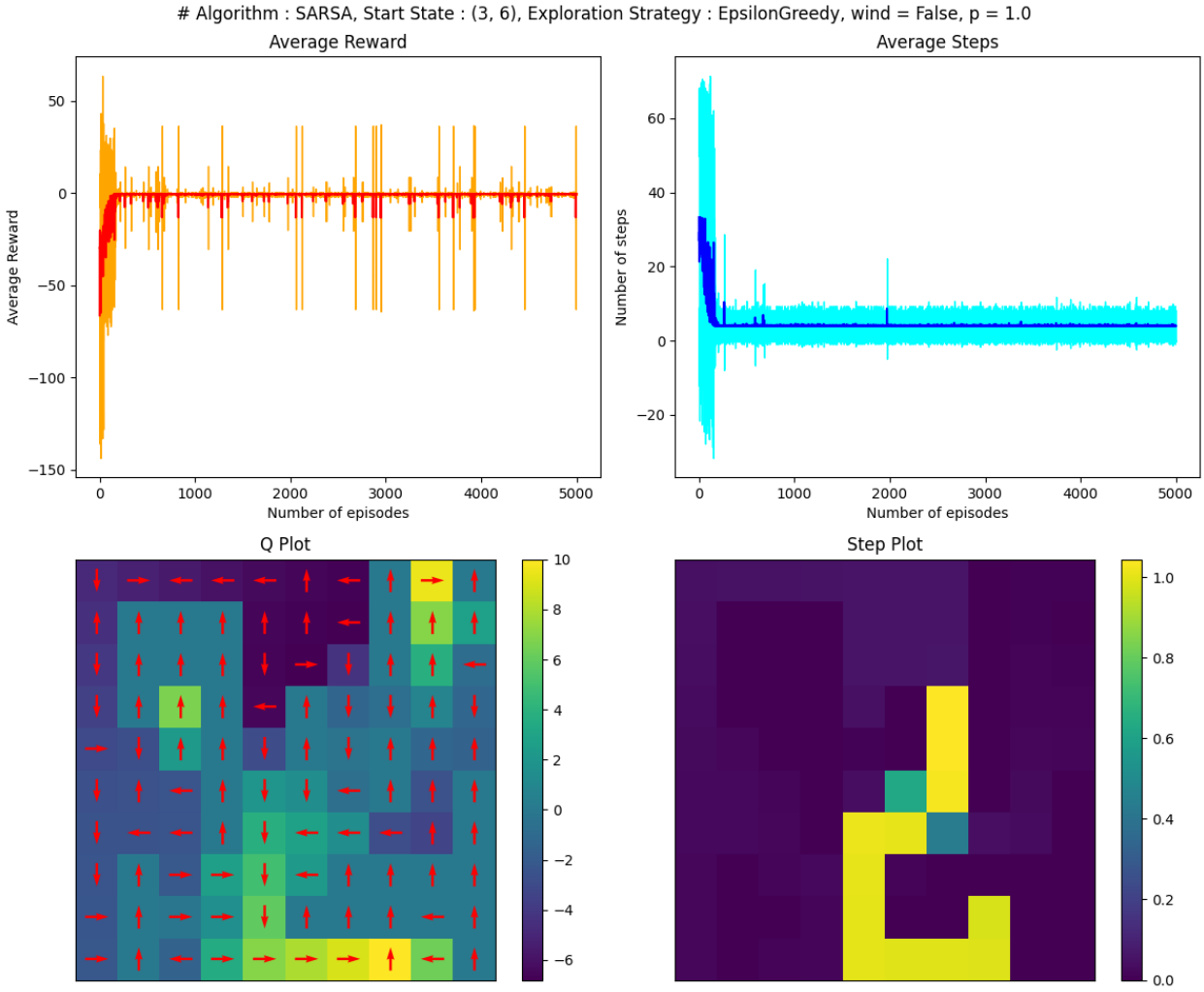

Figure 4: Sarsa behavior, start_state=(3,6), wind=False(p1.0)

From the state-visit count heat-map it is clear that the agent visits the bottom right corner goal state much more frequently than the other two goal states. So it has learnt to reach the bottom right goal state.

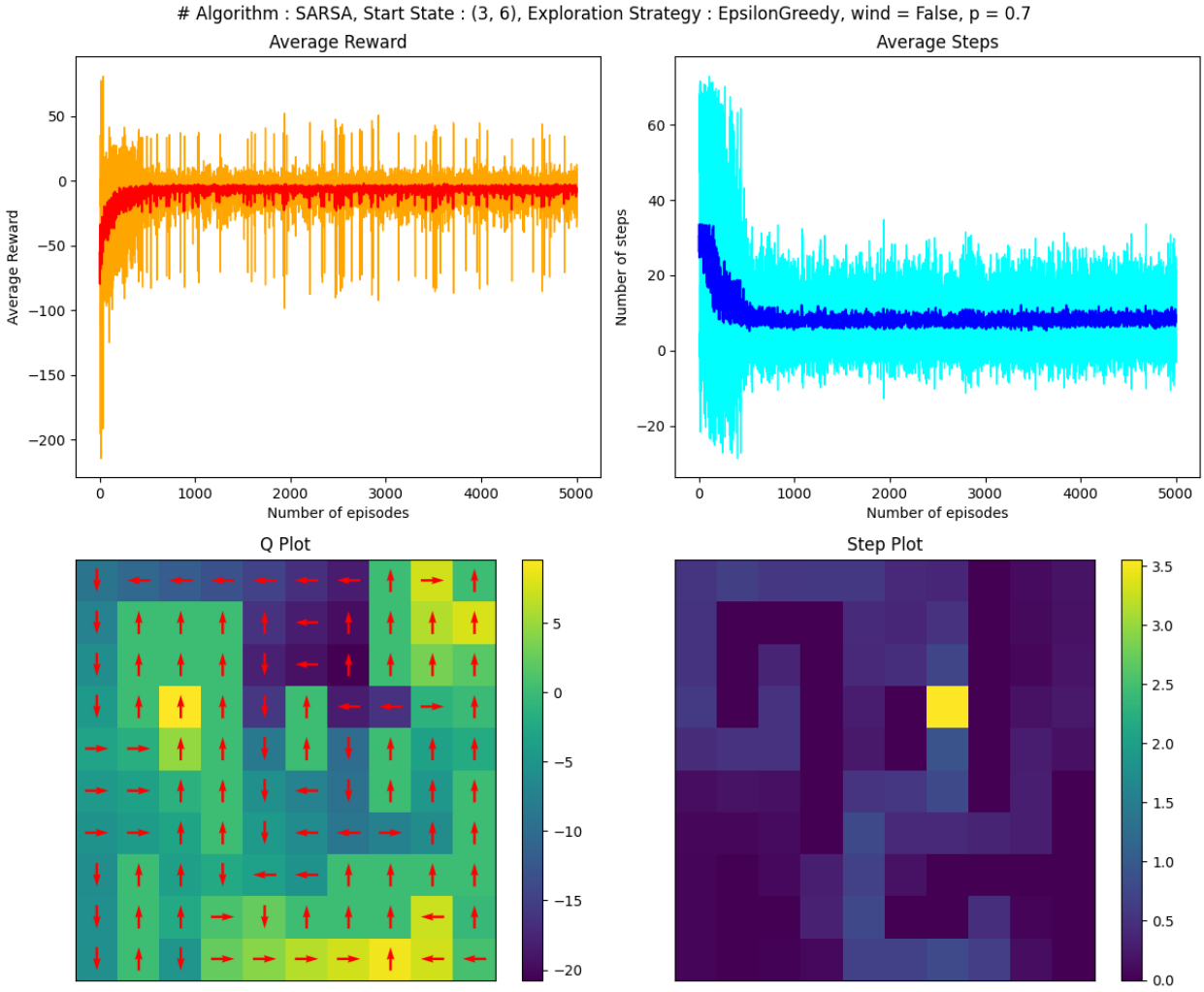

Figure 5: Sarsa behavior, start_state=(3,6), wind=False(p$\bar{0}$.7)

From the state-visit count heat-map it is clear that the agent does not visit any of the three goal states very frequently. However it does visit the bottom right goal state more times than the other two goal states.

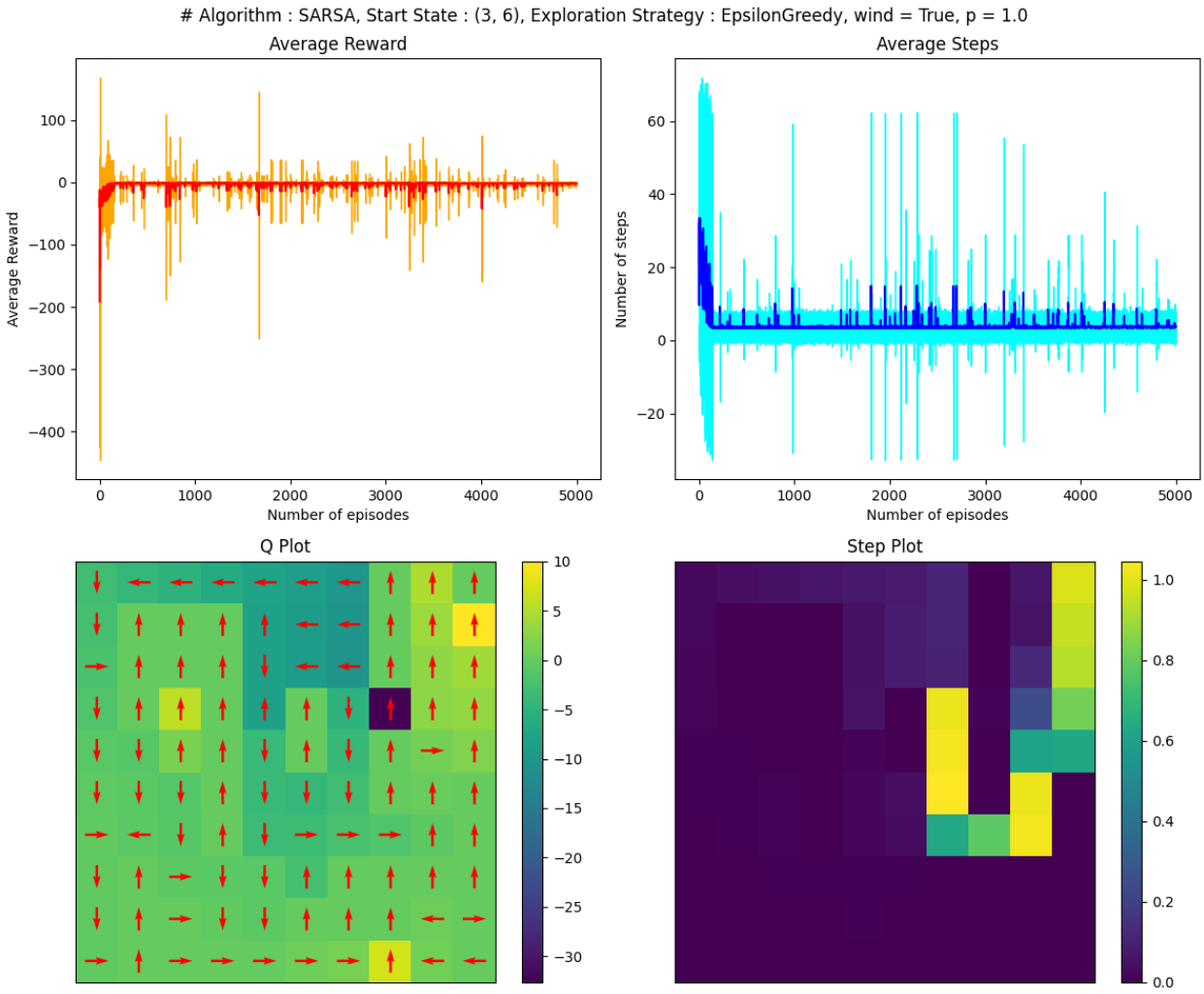

Figure 6: Sarsa behavior, start_state=(3,6), wind=True(p1.0)

From the state-visit count heatmap it is clear that the agent visits top right goal state much more frequently than the other two goal states. It rarely visits the other two goal states. So the agent has learnt to reach the upper right corner goal state.

0.4.1 Sarsa: Tuning of hyper params

The following hyperparameters were tested and experimented out.

- action selection functions = [softmax, epsilon greedy]
- action selection param ϵgreedy = [0.1, 0.01, 0.001]
- action selection param softmax(τ) = [1, 0.1, 0.01]
- gammas = [1, 0.9, 0.8]
- learning rate = [1, 0.1, 0.01]

Below are the results obtained against the best set of params.

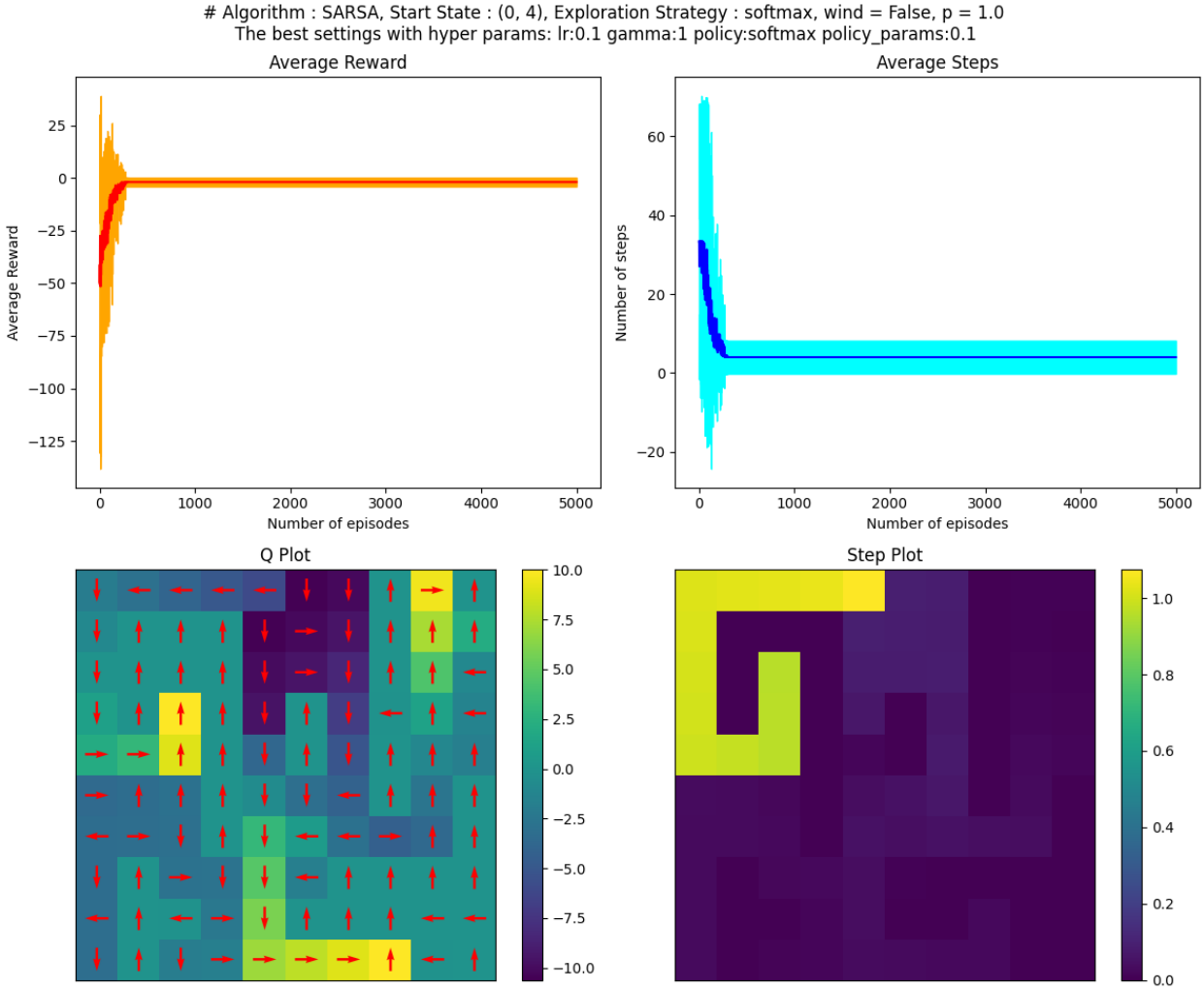

Figure 7: Sarsa behavior, start_state=(0,4), wind=False(p$\bar{1}$.0)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 1
- Policy (π): softmax
- policy param (τ): 0.1

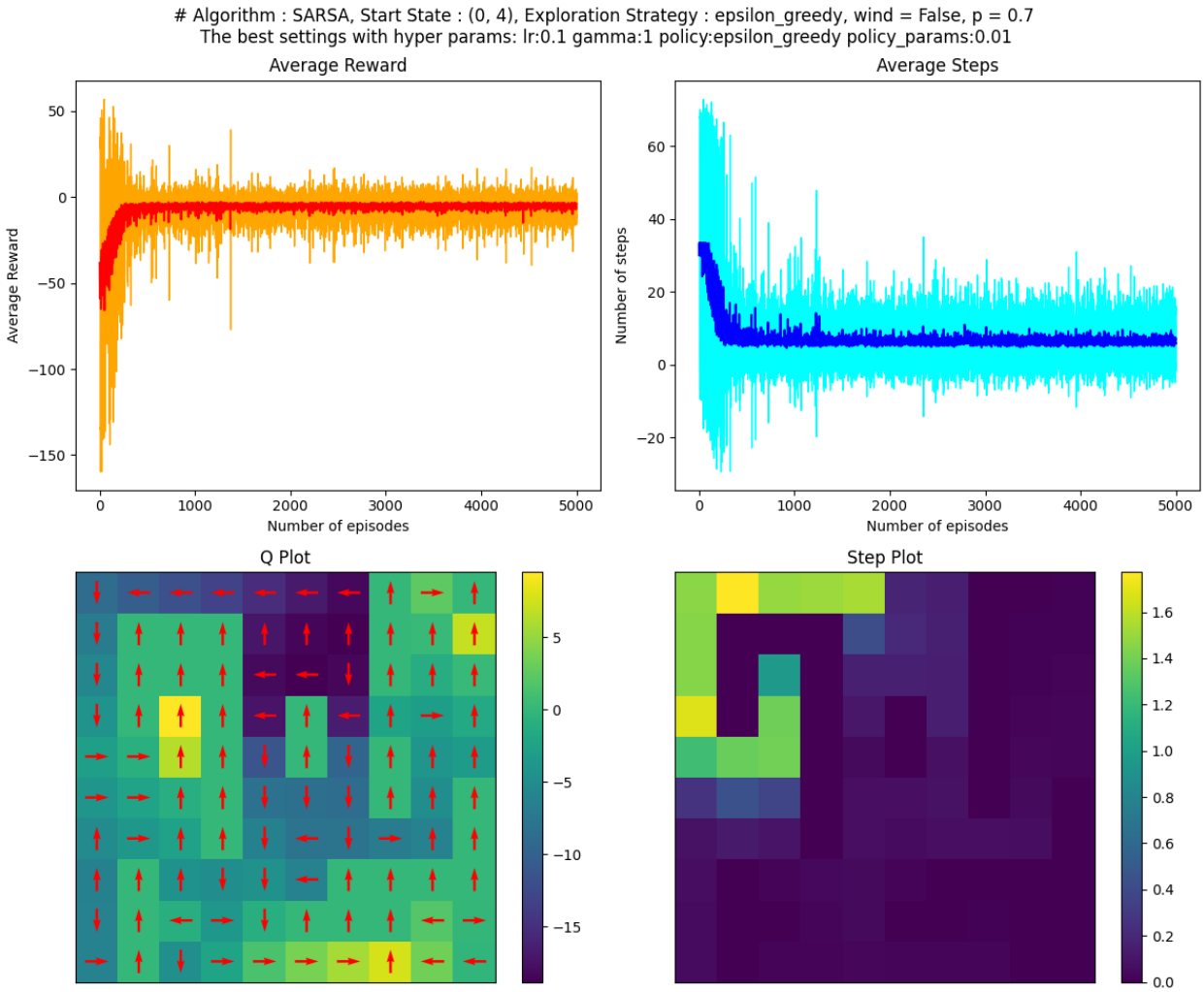

Figure 8: Sarsa behavior, start_state=(0,4), wind=False(p$\bar{0}$.7)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 1
- Policy (π): epsilon greedy
- policy param (ϵ): 0.01

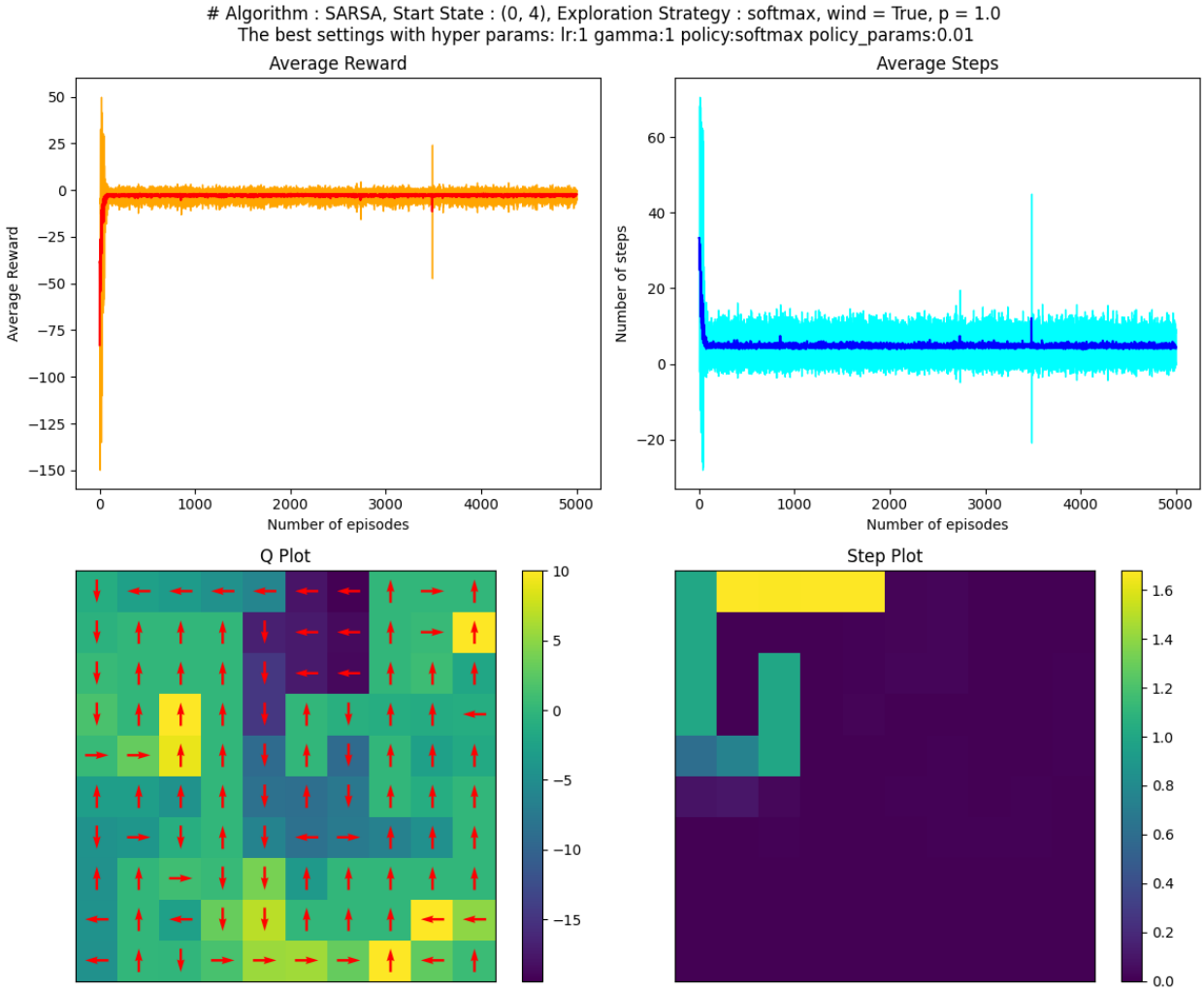

Figure 9: Sarsa behavior, start_state=(0,4), wind=True(p1.0)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 1
- Policy (π): softmax
- policy param (τ): 0.1

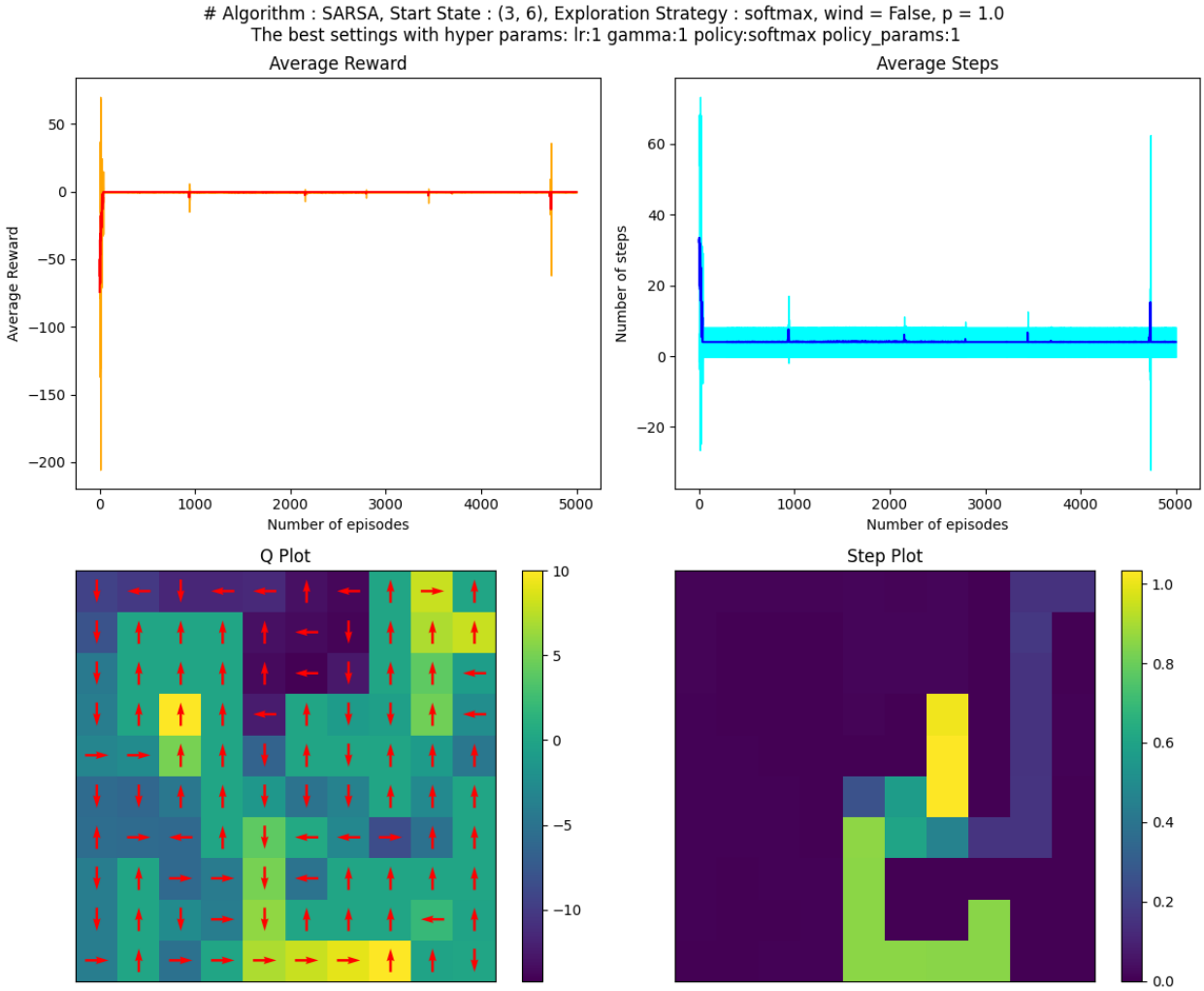

Figure 10: Sarsa behavior, start_state=(3,6), wind=False(p1.0)

Best Hyper parameter settings

- Learning rate (α): 1
- Discount rate (γ): 1
- Policy (π): softmax
- policy param (τ): 1

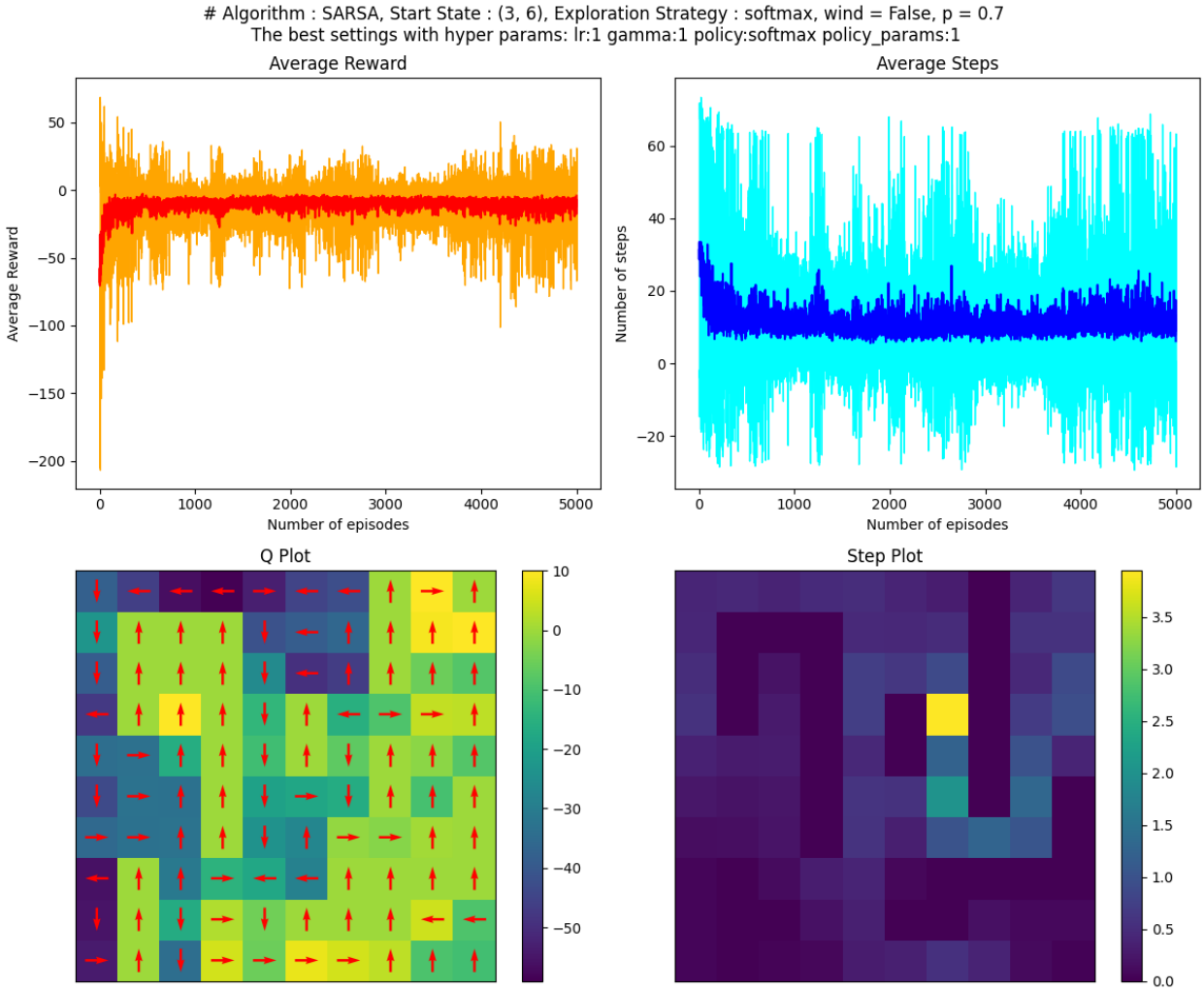

Figure 11: Sarsa behavior, start_state=(0,4), wind=False(p0.7)

Best Hyper parameter settings

- Learning rate (α): 1
- Discount rate (γ): 1
- Policy (π): softmax
- policy param (τ): 1

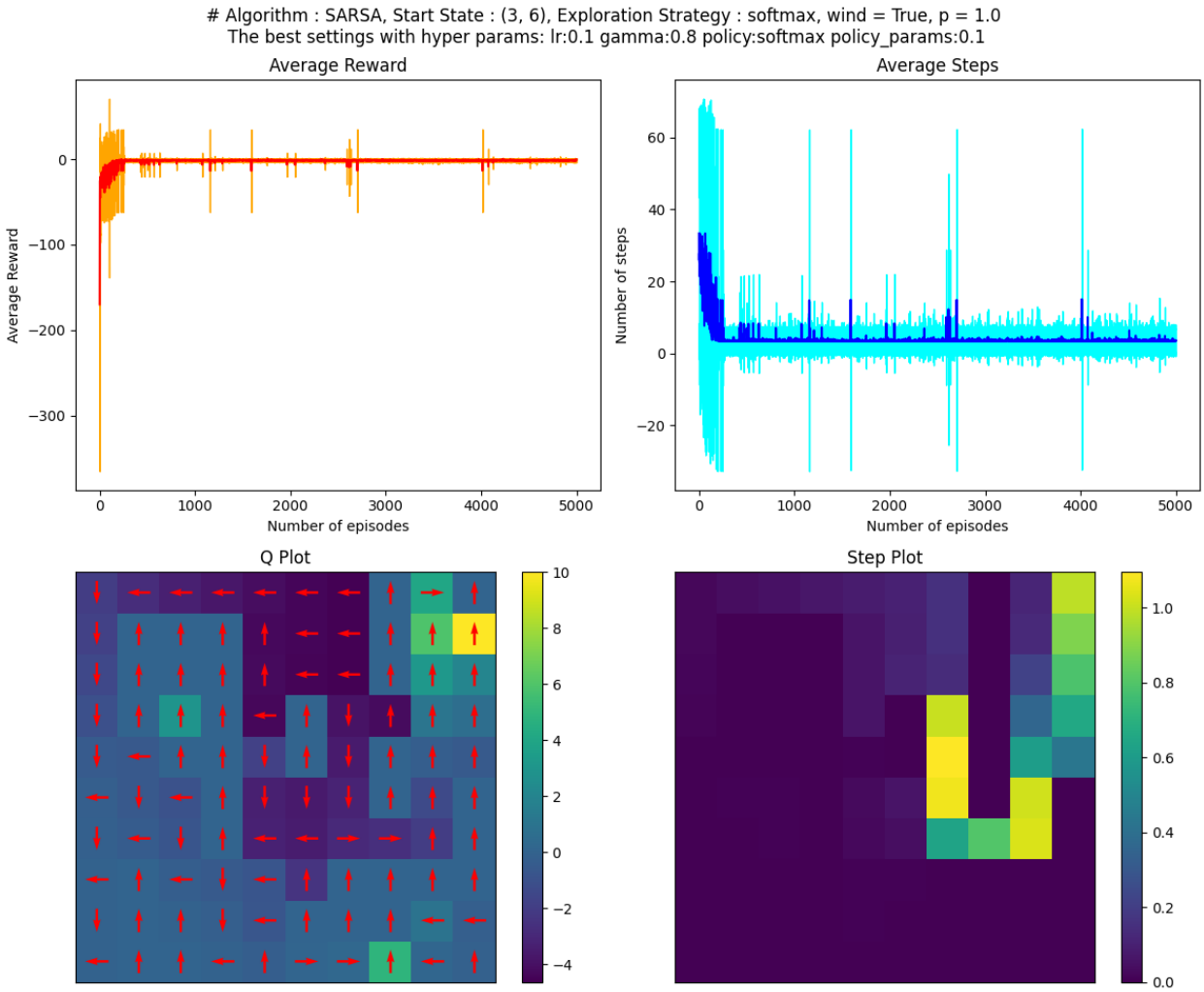

Figure 12: Sarsa behavior, start_state=(0,4), wind=True(p1.0)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 0.8
- Policy (π): softmax
- policy param (τ): 0.1

0.5 Behavior of Q learning off policy algorithm [1]

Algorithm 2 Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

States $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
    Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
    while Q is not converged **do**
        Start in state $s \in \mathcal{X}$
        while s is not terminal **do**
            Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow_a Q(x, a)$)
            $a \leftarrow \pi(s)$
            $r \leftarrow R(s, a)$ ▷ Receive the reward
            $s' \leftarrow T(s, a)$ ▷ Receive the new state
            Calculate π based on Q (e.g. epsilon-greedy)
            $s \leftarrow s'$
        end while
    end whilereturn Q
end procedure

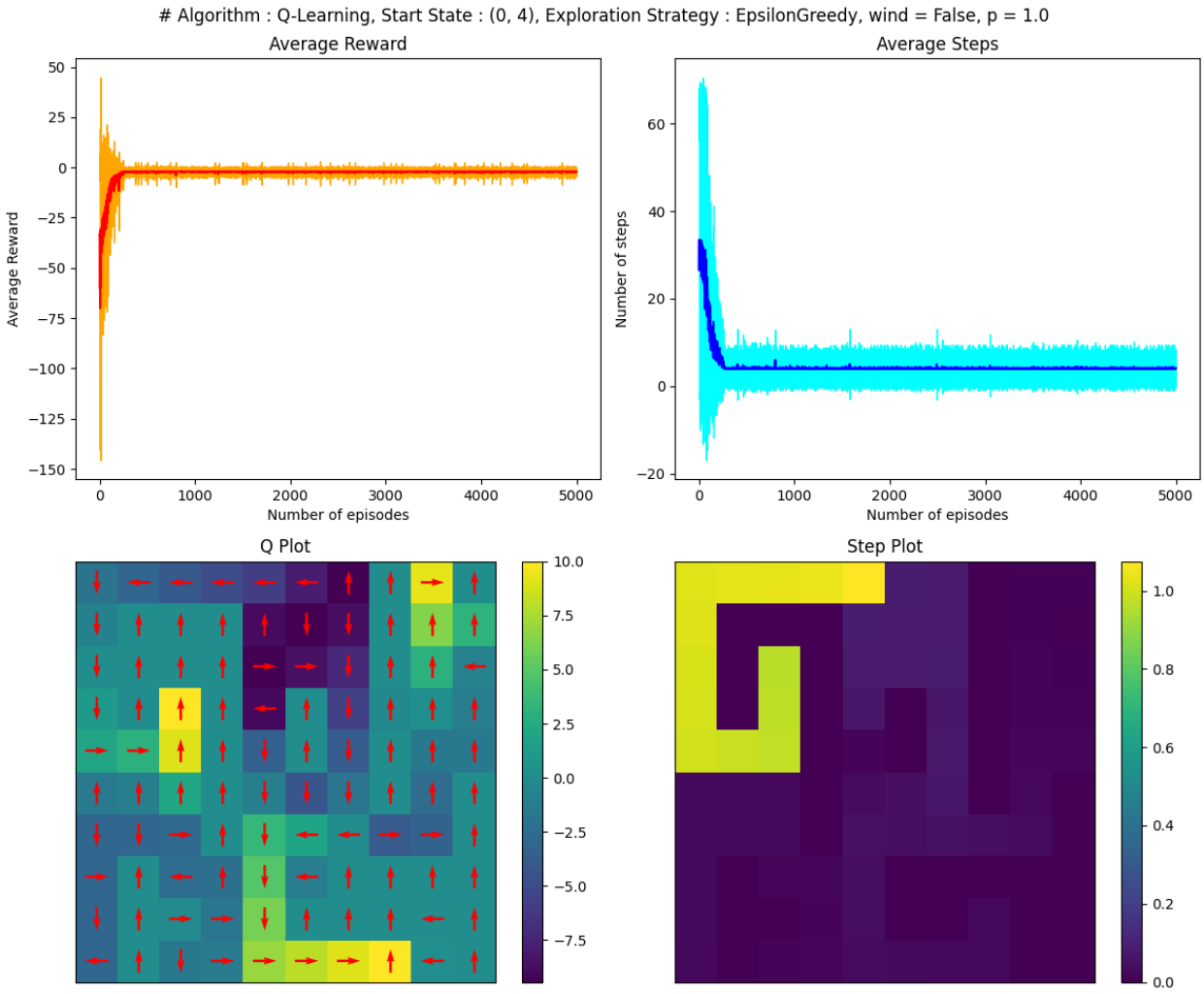

Figure 13: Q learning, start(0,4), windFalse(p=1.0)

From the state-visit count heat-map it can be seen that the agent visits the upper left corner very frequently. It has learnt to reach the upper left corner goal state.Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

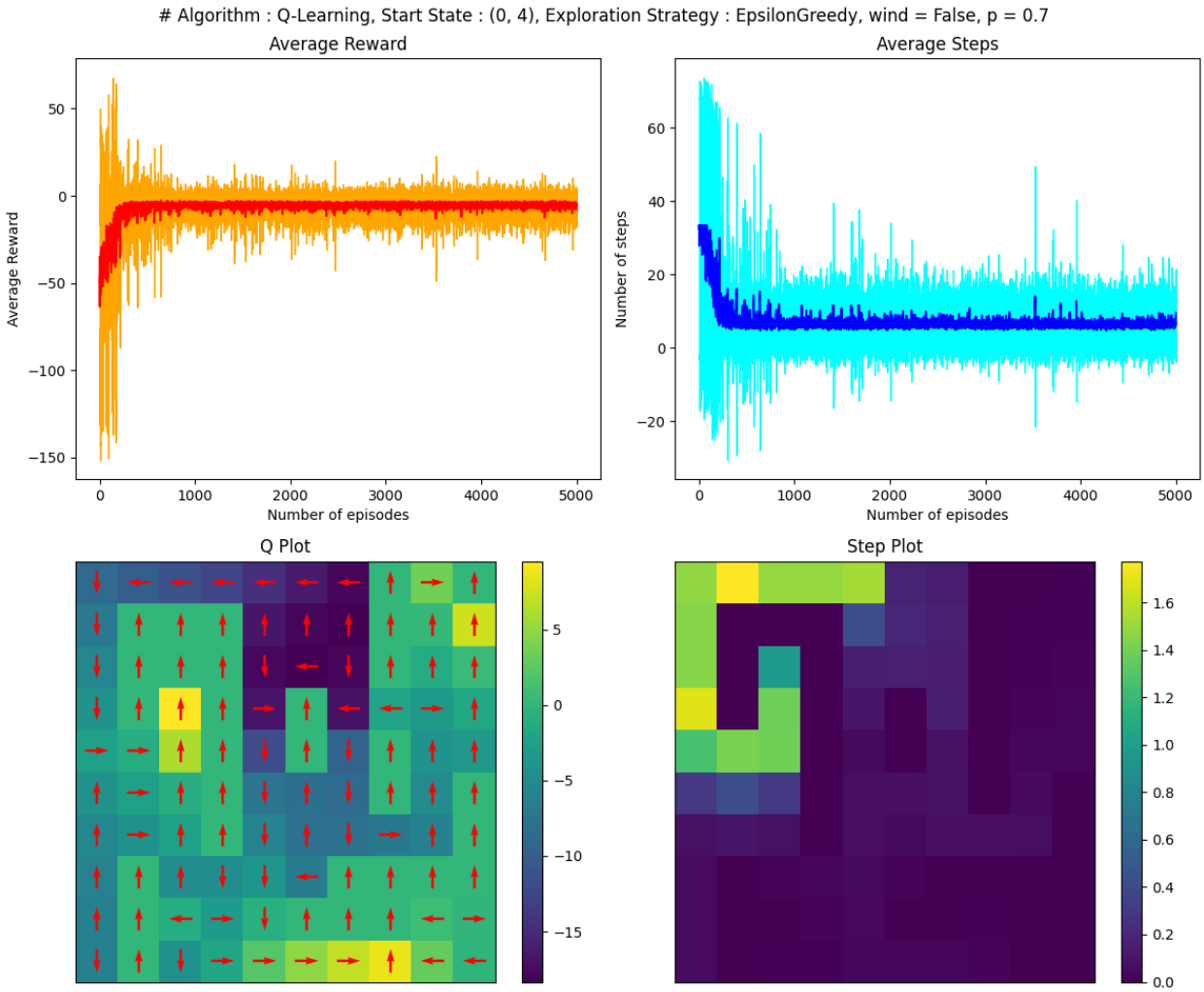

Figure 14: Q learning, start(0,4), windFalse(p=0.7)

From the state-visit count heat-map it can be seen that the agent visits the upper left corner very frequently. Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

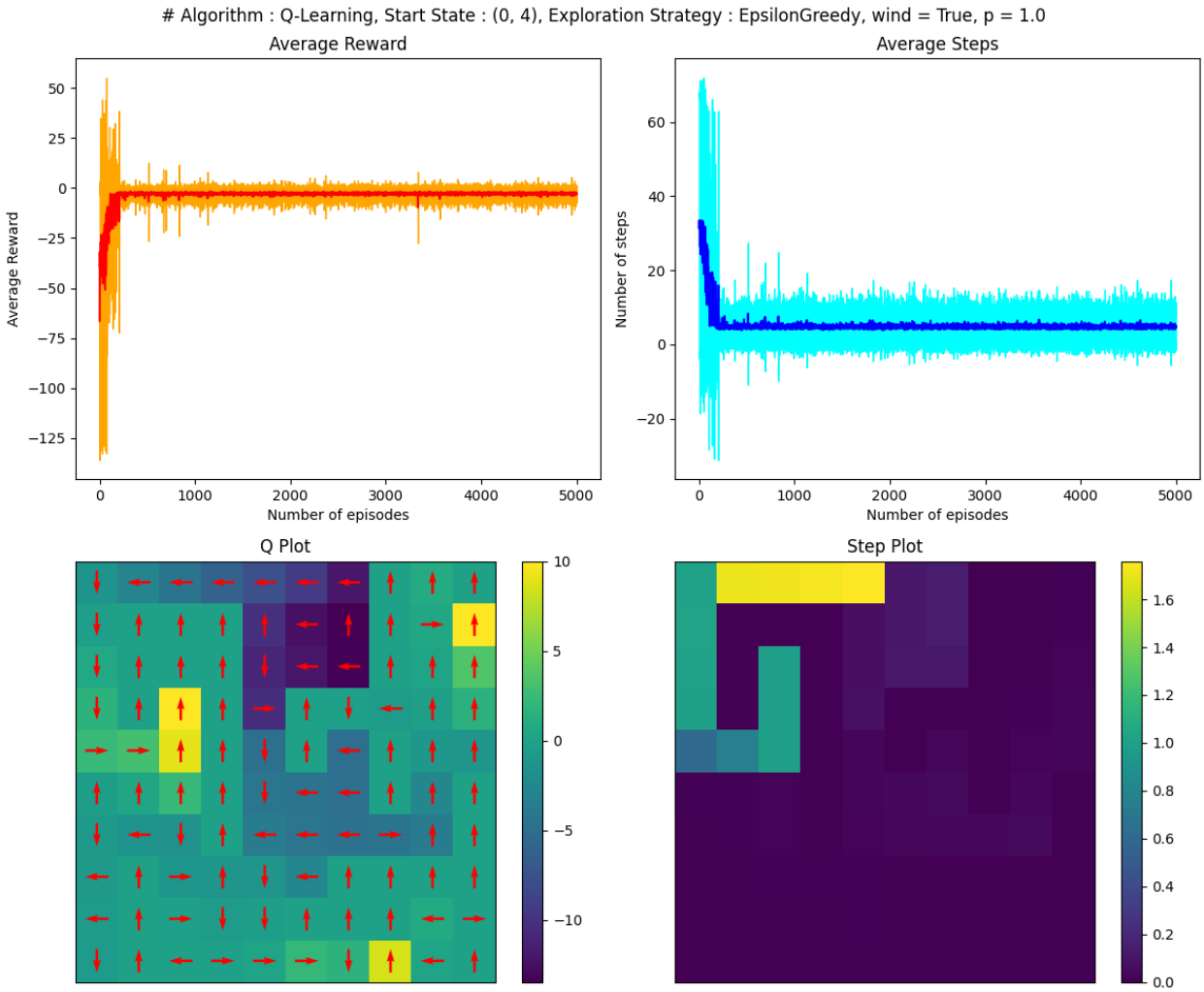

Figure 15: Q learning, start(0,4), windTrue(p=1.0)

From the state-visit count plot it is clear that the agent visits the upper left corner frequently. Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

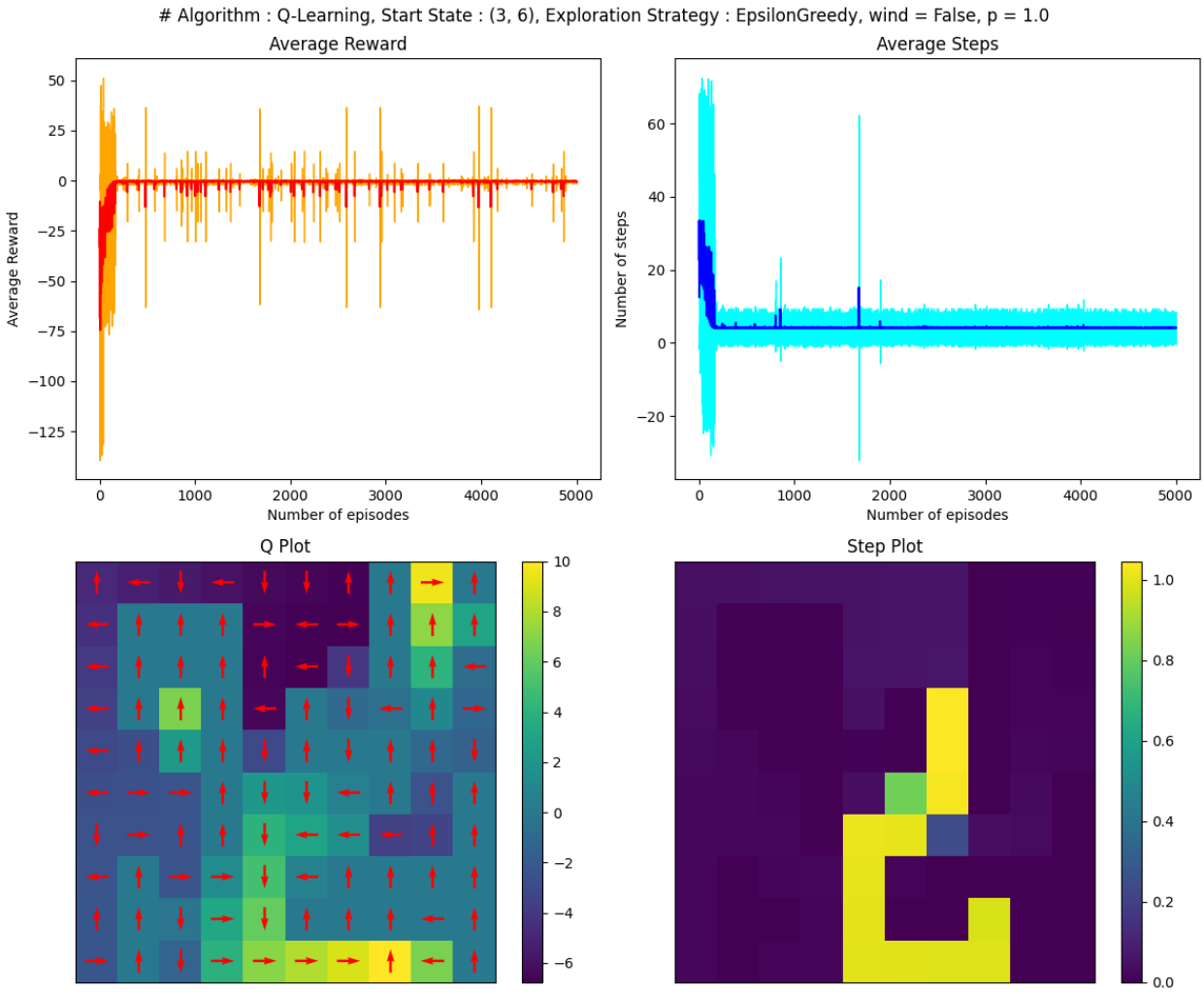

Figure 16: Q learning, start(3,6), windFalse(p=1.0)

From the state-visit count plot it is clear that the agent has learnt to reach the bottom mid goal state. Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

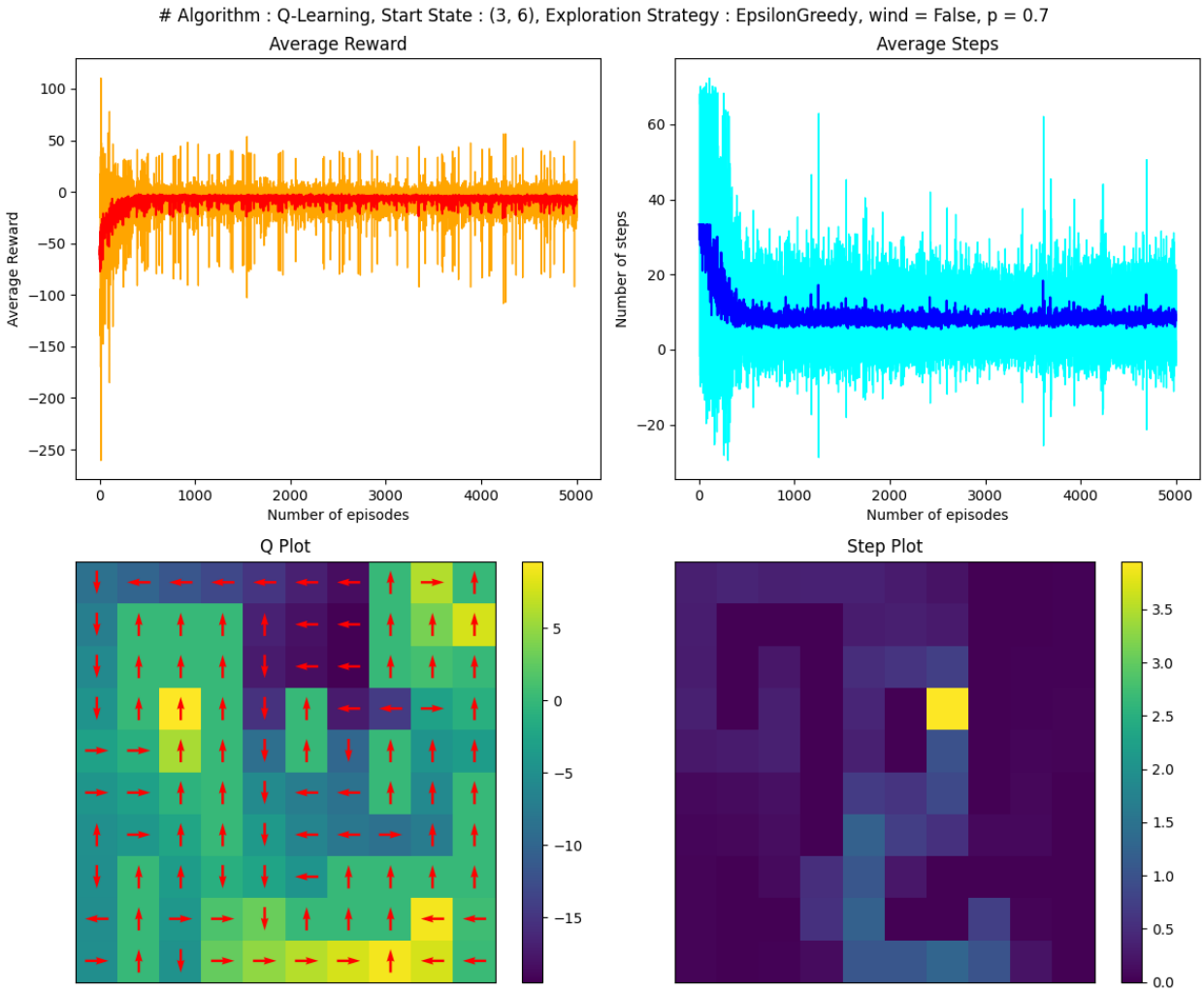

Figure 17: Q learning, start(3,6), windFalse(p=0.7)

From the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high. High stochasticity of the environment lead to a noisy estimate. And also we see from the state visit count that, the agent tried to reach the two goals, one in the upper left corner and bottom right corner.But it is evident that the state visit count near the bottom right corner is quite high vs the upper left corner.

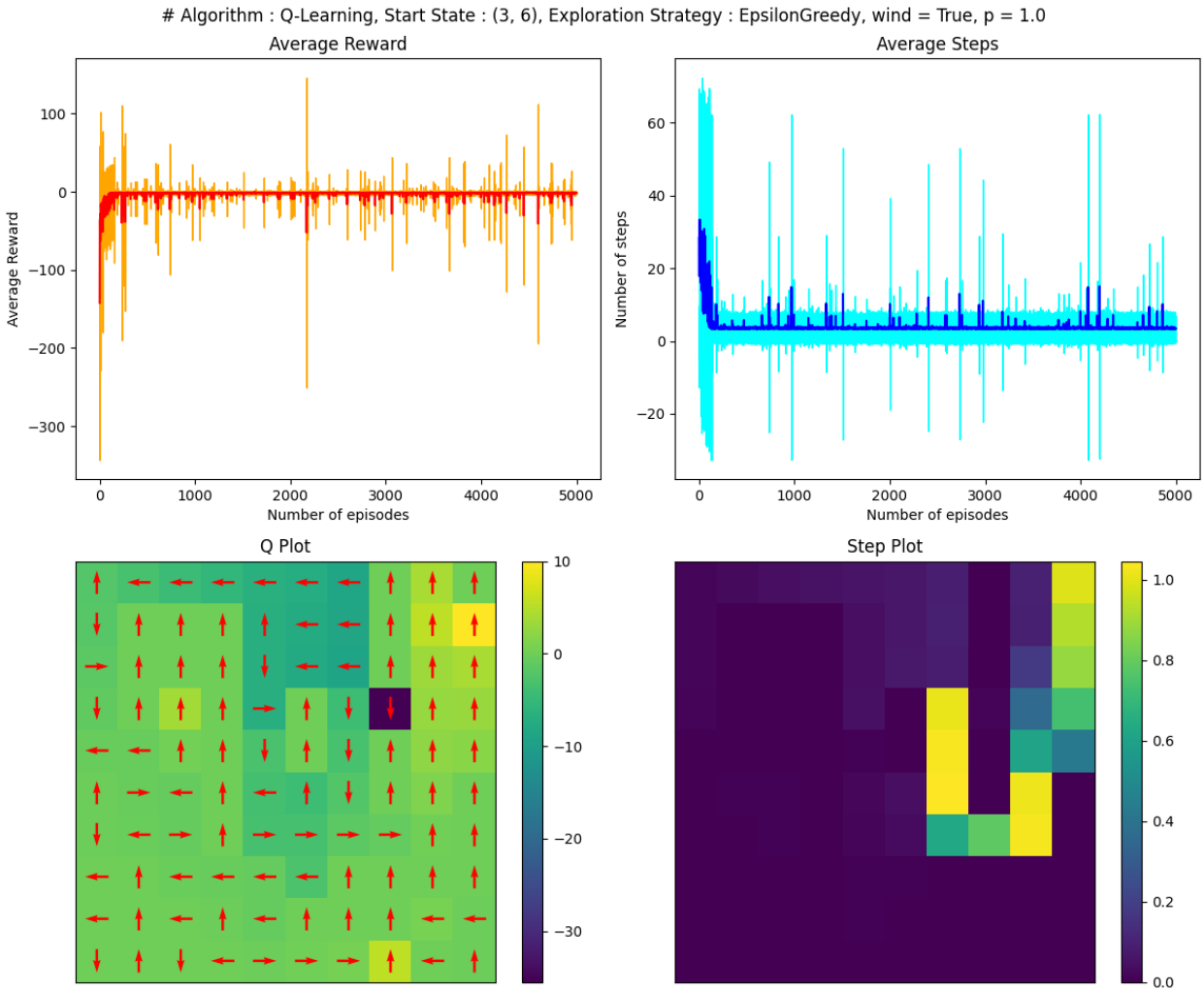

Figure 18: Q learning, start(3,6), wind True(p=1.0)

From the state-visit count plot it is clear that the agent visits the mid upper frequently. Also from the Q-value heat-map it is clear that the Q-values near the goal states(for the optimal actions) are high.

0.5.1 Q Learning: Tuning of hyper params

The following hyperparameters were tested and experimented out.

- action selection functions = [softmax, epsilon greedy]
- action selection param ϵgreedy = [0.1, 0.01, 0.001]
- action selection param softmax(τ) = [1, 0.1, 0.01]
- gammas = [1, 0.9, 0.8]
- learning rate = [1, 0.1, 0.01]

Below are the results obtained against the best set of params.

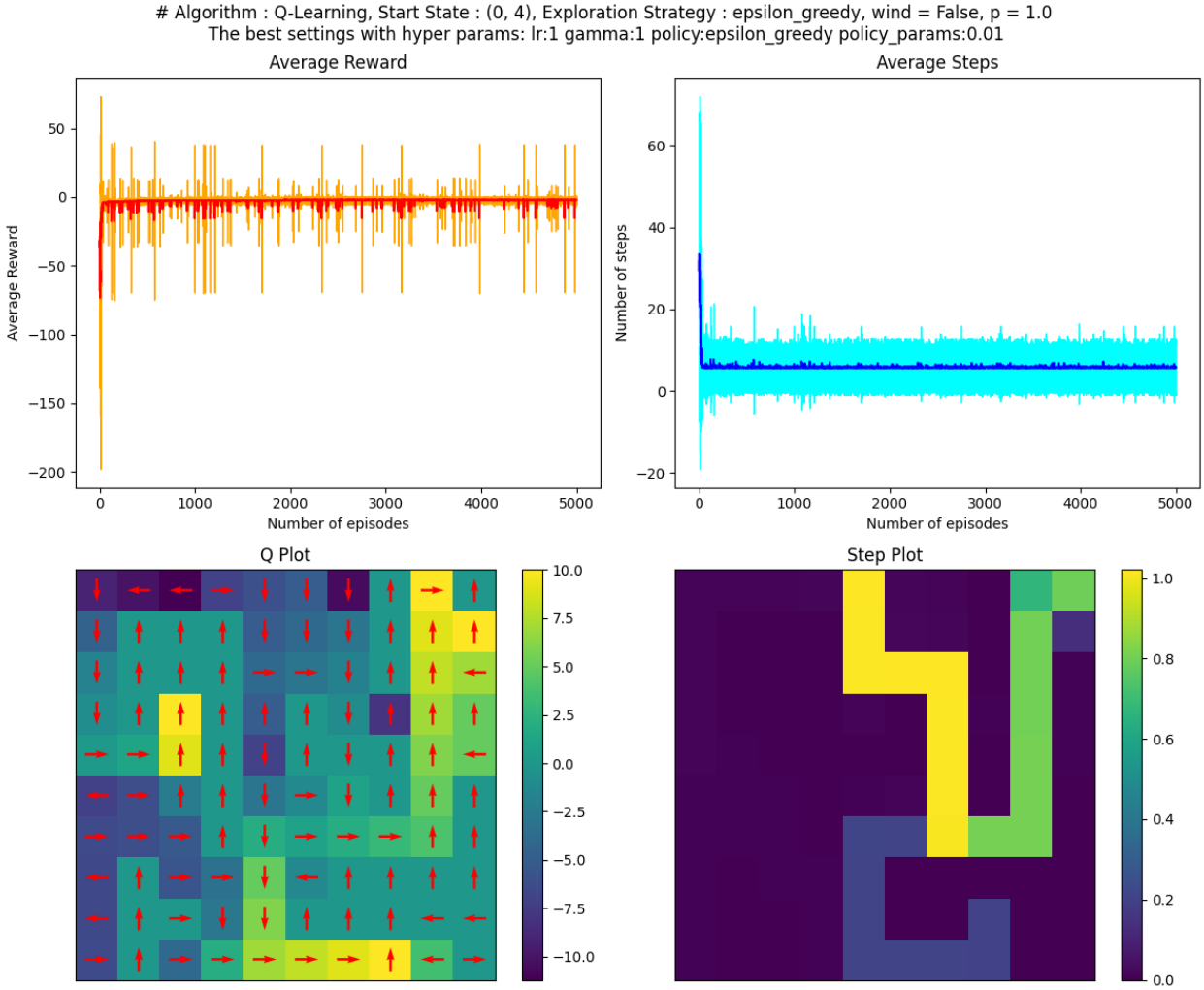

Figure 19: Q learning, start(0,4), windFalse(p=1.0)

Best Hyper parameter settings

- Learning rate (α): 1
- Discount rate (γ): 1
- Policy (π): epsilon greedy
- policy param (ϵ): 0.01

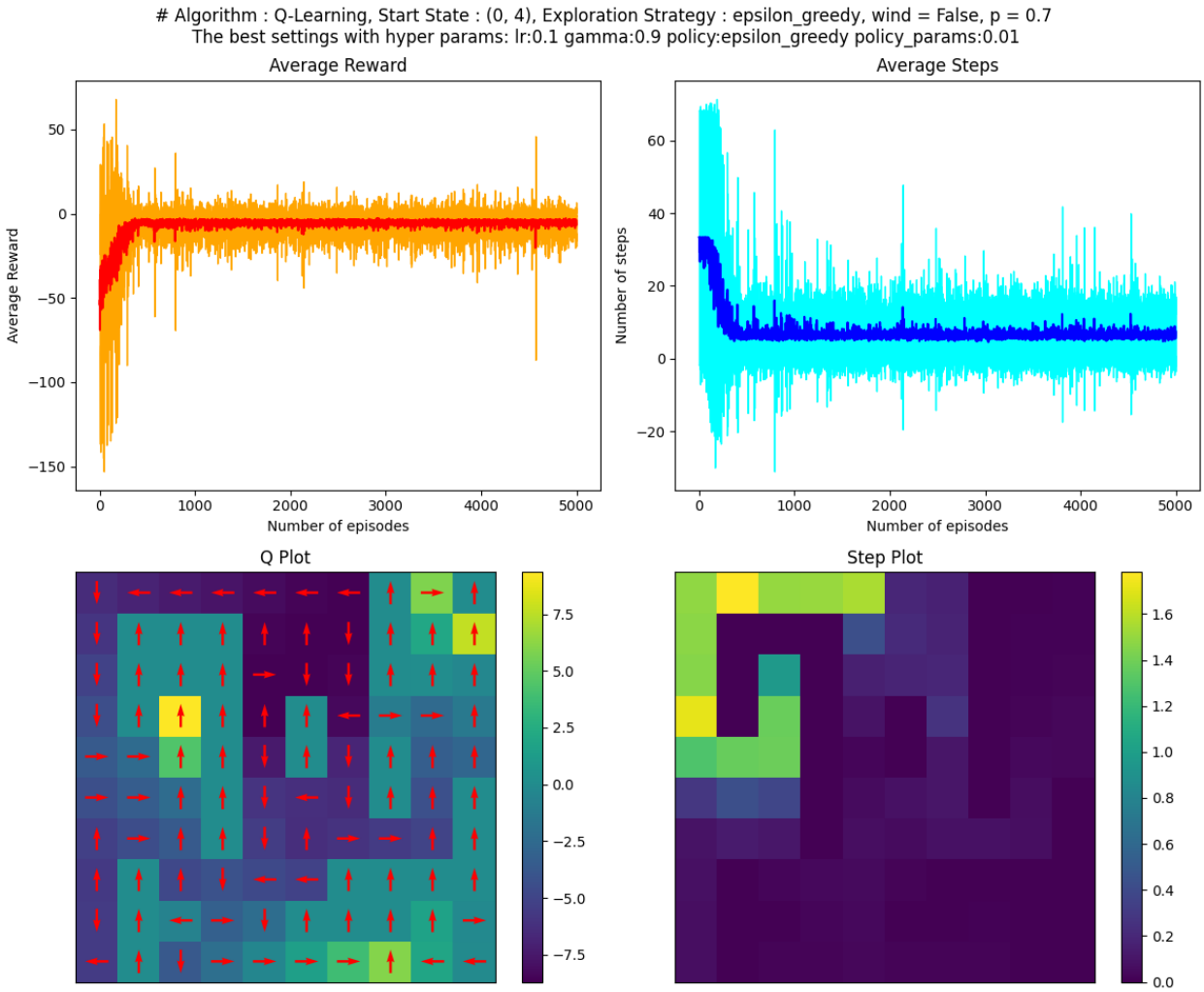

Figure 20: Q learning, start(0,4), windFalse(p=0.7)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 0.9
- Policy (π): epsilon greedy
- policy param (ϵ): 0.01

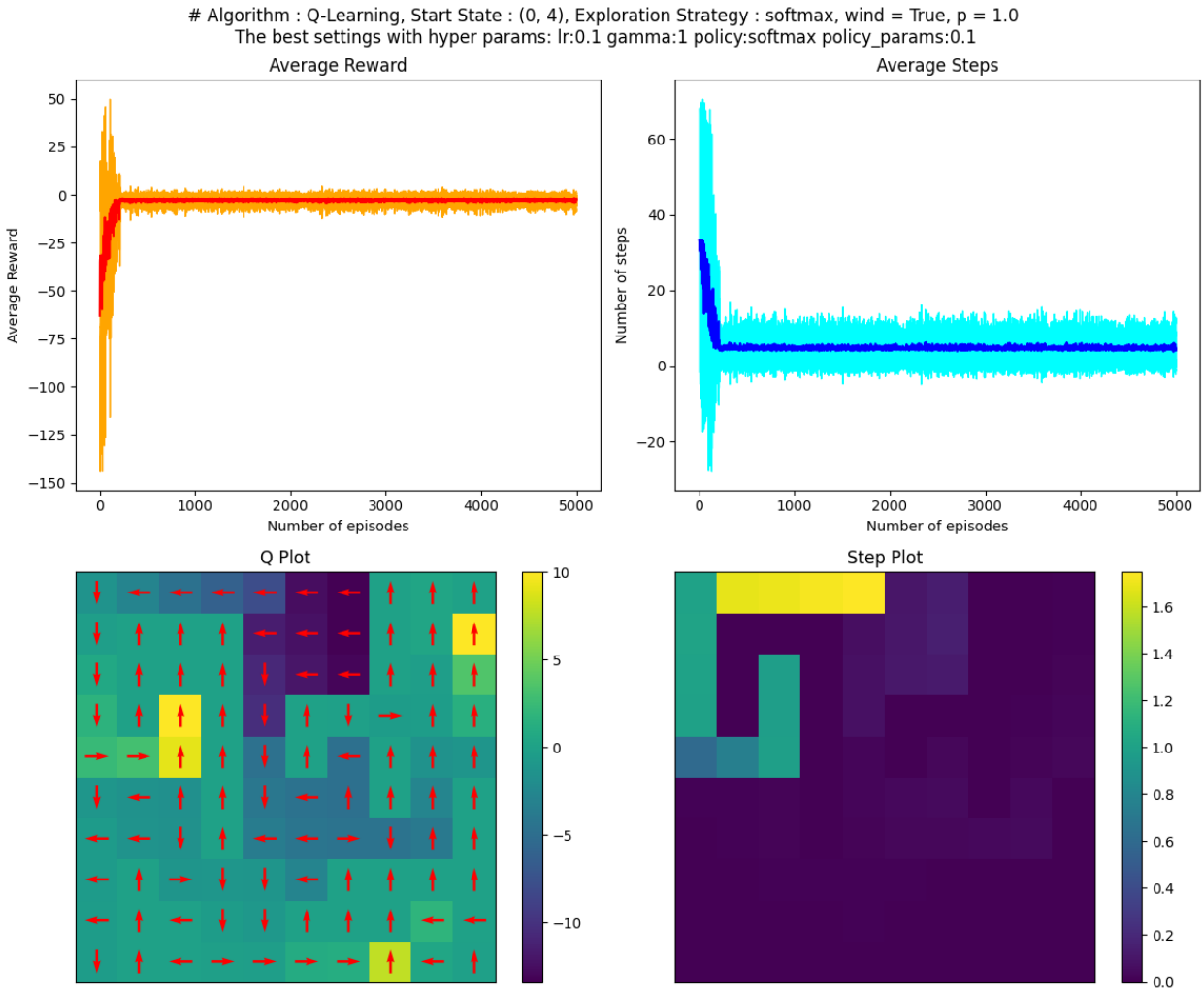

Figure 21: Q learning, start(0,4), wind True(p=1.0)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 1
- Policy (π): softmax
- policy param (τ): 0.1

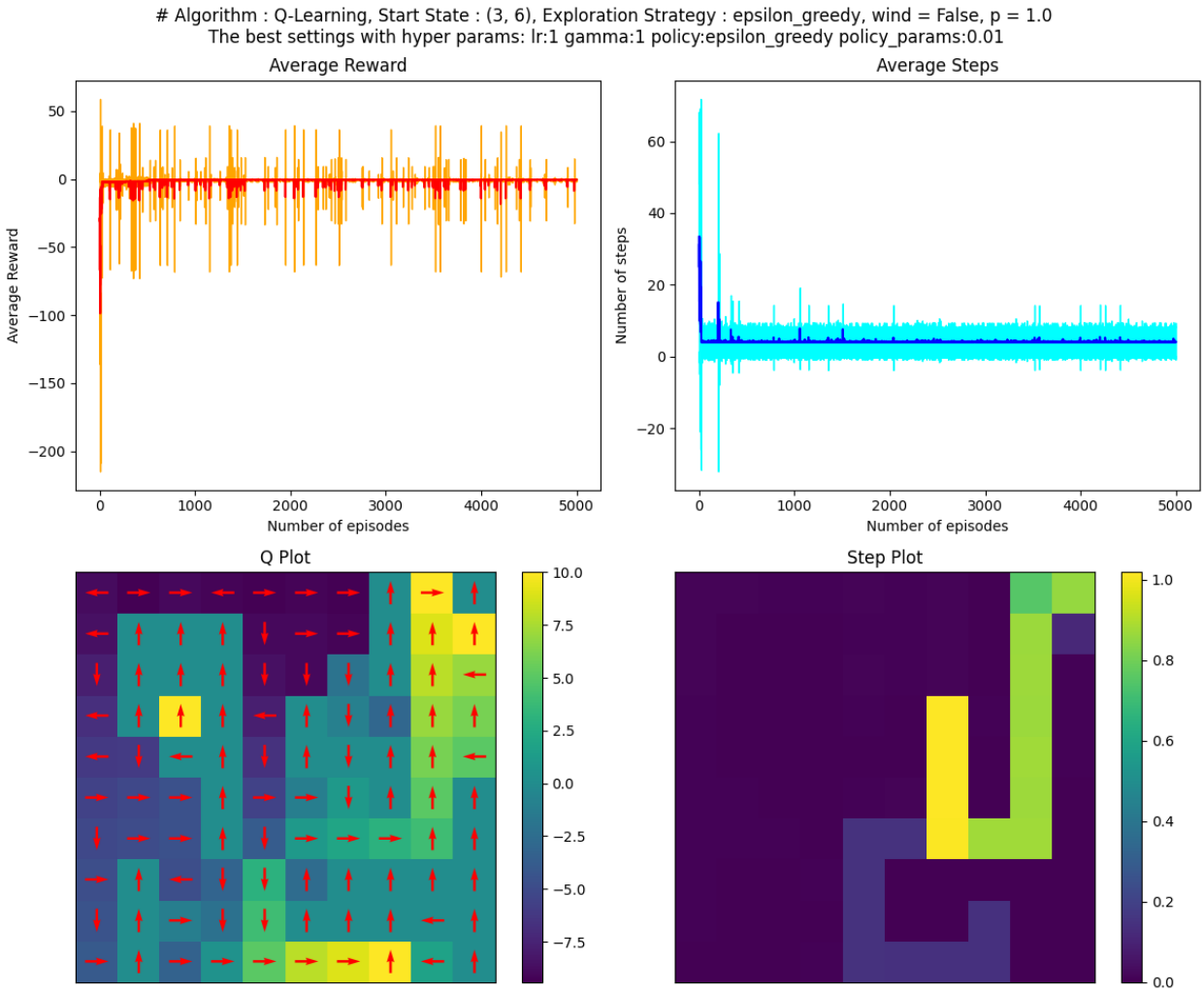

Figure 22: Q learning, start(3,6), windFalse(p=1.0)

Best Hyper parameter settings

- Learning rate (α): 1
- Discount rate (γ): 1
- Policy (π): epsilon greedy
- policy param (ϵ): 0.01

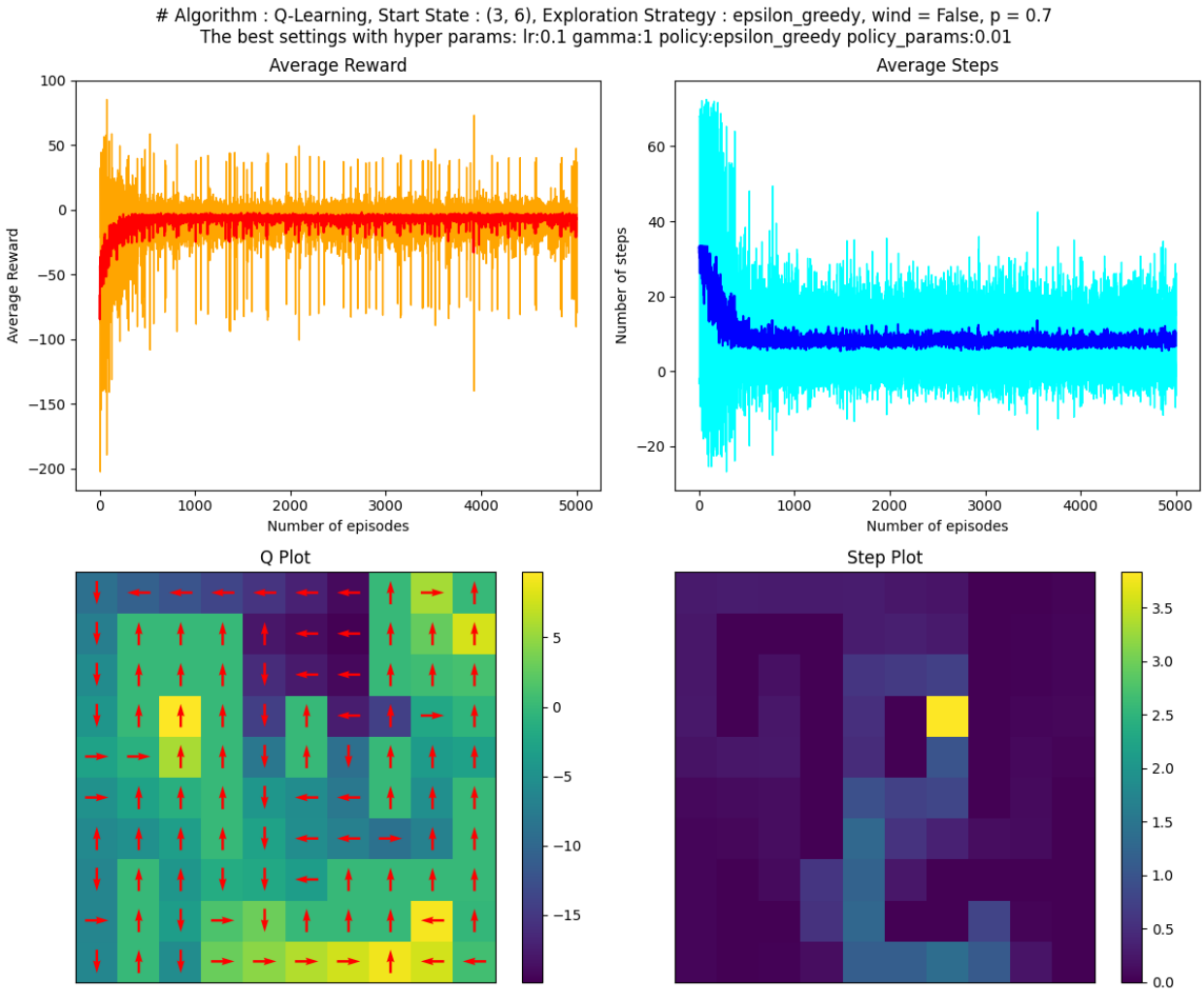

Figure 23: Q learning, start(3,6), windFalse(p=0.7)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 1
- Policy (π): epsilon greedy
- policy param (ϵ): 0.01

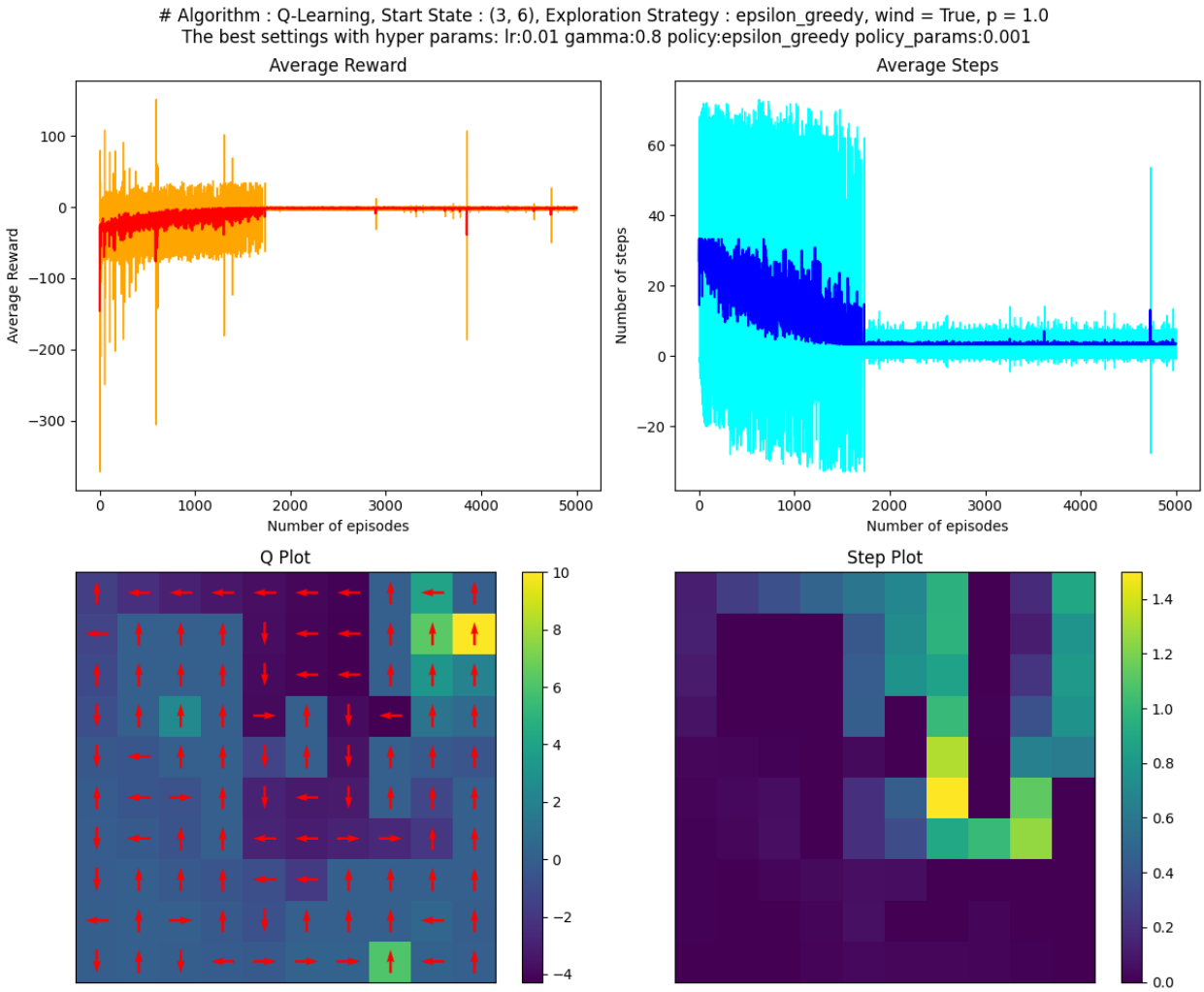

Figure 24: Q learning, start(3,6), wind True(p=1.0)

Best Hyper parameter settings

- Learning rate (α): 0.1
- Discount rate (γ): 0.8
- Policy (π): epsilon greedy
- policy param (ϵ): 0.001

Bibliography

[1] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, PP:1–1, 09 2019.

[2] G. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.