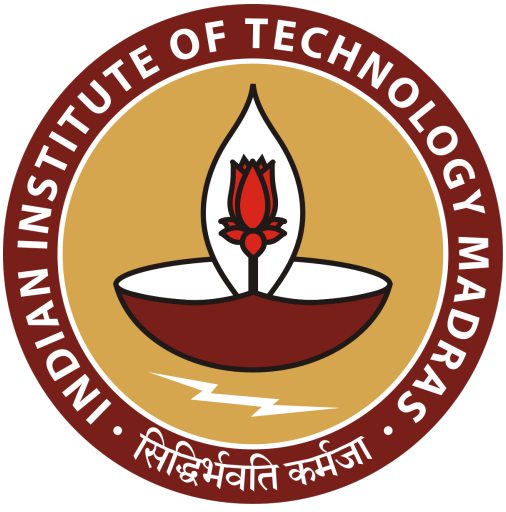
Programming Assignment 1

SARSA and Q-Learning

Gautham Govind A - EE19B022
Vishnu Vinod - CS19B048

CS6700
Reinforcement Learning



Department of Computer Science and Engineering
IIT Madras

Contents

1 CODE FOLDER

The entirety of the code written for this assignment as well as the log files generated during hyperparameter tuning and the output files containing all plots and heatmaps are present in this Google Drive Link: `https://drive.google.com/drive/folders/1nAHkAv6-zYXTZmjq00XhuOXNO0Zes_Wx?usp=sharing`

2 Introduction

In this assignment we study two important methods based on Temporal Difference (TD) Learning, namely: **SARSA** and **Q-Learning**. Temporal Difference Learning is the name given to a class of model-free RL methods. They learn by bootstrapping better estimates of the value function(s) by using the current estimates and improving on it gradually. TD Learning can be roughly seen as a blend of both Monte Carlo methods and Dynamic Programming (DP). The most important features of TD Learning can be listed as follows:

- It is a **model-free** method. It does not require knowledge of the model state transition probabilities.
- The RL agent learns from **sampled experience** like various Monte Carlo methods
- TD methods **bootstrap** (like various DP) based on learned estimates without waiting for the final outcome or end of the episode (for episodic tasks).
- Since they **learn from incomplete episodes** too, and are used in continuous (non-episodic) tasks.

The two methods are used to solve different variants of the Grid-World problem described below.

3 Environment - All the World's a Grid

The environment described for this assignment is a modification of the classic Grid-World problem that allows us to test out multiple variants of the problem by changing a few parameters. The following is the example of the grid world (as given in the problem statement):

We are given a 10×10 gridworld which has obstacles and end states as shown in the figure. There are two possible start states of which only one can be present at a time. It also has a couple of other states which are discussed below.

3.1 Types of States

There are a total of 6 types of states, seen in the below image:

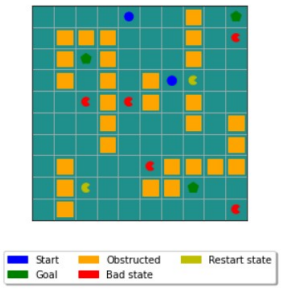


Figure 1: Grid World

- **Start State**: The agent starts out from this state. ([0,4] and [3,6] are the two possible start states we consider)
- **Goal State**: The goal is to reach one of these states. There are 3 goal states in total. They give high reward on entry (+10)

- **Normal State**: These are normal states which give a small step penalty while entering (to give incentive for agent to learn shortest path)
- **Obstructed State**: These are walls that prevent entry to the respective cells. Transition to these states will not result in any change.
- **Bad State**: Entry into these states will incur a higher penalty than a normal state (-6 compared to a -1 penalty for normal states).
- **Restart State**: Entry into these states will incur a very high penalty(-100) and will cause agent to teleport to the start state without the episode ending.

3.2 Actions: To Move or Not to Move

This is a grid world with 4 deterministic actions ('UP', 'DOWN', 'LEFT', 'RIGHT'). The agent transitions to the next state determined by the direction of the action chosen, with a probability of $p \in [0, 1]$. Let this action direction be designated 'NORTH'. The $1 - p$ probability is divided into the two directions perpendicular to the chosen action namely 'EAST' and 'WEST' with the help of another parameter called bias, $b \in [0, 1]$. The agent transitions to the state West of the chosen action with probability $(1 - p) \times b$, and to the East of the chosen action with probability $(1 - p) \times (1 - b)$.

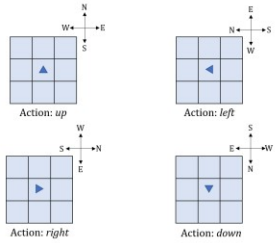


Figure 2: Actions

It is important to note here that the value of the bias probability b we have decided to use is 0.5. Although this value isn't provided in the problem statement, we used this value since this is the default value provided in the starter code.

3.3 Winds of Change

In addition to the stochasticity introduced into the Grid-world problem, we also have the optional presence of a wind. When the wind is present, it may push the agent one additional cell to the right with a probability $p_w = 0.4$. This wind direction is fixed with respect to the grid world - does not depend on the direction of movement of the agent (the action of the wind only takes place **after the agent's move is completed)**.

3.4 Optimality: A Human Perspective

Before training a reinforcement learning agent on the environment, it might be worthwhile to take a pause and see how a human would solve the problem. Given the fact that the environment is rather small, it is not so difficult to see what would be the optimal policies for different start states, at least in the deterministic setup. These are discussed below:

3.4.1 Start State: [0, 4]

From this start state, it so happens that optimal paths to all goal states **incur the same penalty.** Whereas the paths to the goal states at [8, 7] and [0, 9] consists only of normal states, the path to the goal state at

[2, 2], though shorter, necessarily passes through a bad state, thereby cancelling out the effect of the shorter path length. In all three cases, the agent receives a total reward of -6 upon reaching the goal state. The optimal paths are illustrated in Figure 3.

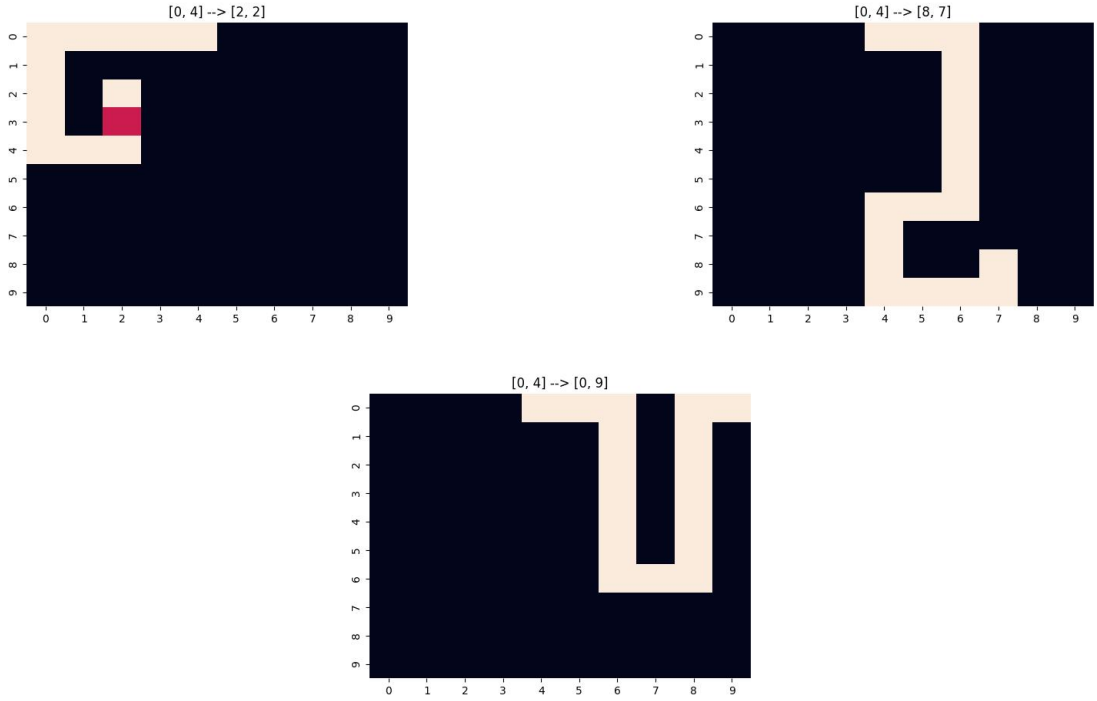


Figure 3: Optimal Paths for the first start position (Red indicates a bad state)

3.4.2 Start State: [3, 6]

From this start state, the optimal paths to the goal states at [8, 7] and [0, 9] incur the same penalty and hence are optimal, whereas the path to the foal state at [2, 2] incurs a higher penalty. The optimal total reward in this case is -1 on reaching the goal state. The optimal paths are illustrated in Figure 4.

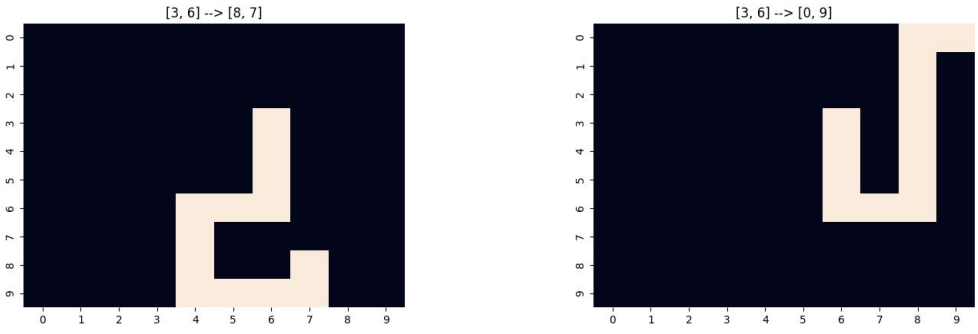


Figure 4: Optimal Paths for the second start position

4 SARSA

SARSA, short for **State Action Reward State Action**, is a Temporal Difference (TD) control algorithm for optimal policy learning. The key idea is to learn the state action value $q_\pi(s, a)$ for all the state action pairs for a policy π and simultaneously update the policy such that it gets closer to an optimal policy π^*.

SARSA essentially implements Generalized Policy Iteration(GPI), but using state action values instead. SARSA is an **On-Policy control algorithm**, in the sense that updates to the action values are done as per the current policy. The SARSA control algorithm has two components: an update rule and an exploration policy.

4.1 SARSA Update Rule

The SARSA update rule is given as follows:

$$\boxed{Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]} \tag{1}$$

The update rule can be interpreted as follows: The state action value, which represents the expected return, is updated each time we obtain a reward. Each reward can be thought of as a sample from the reward distribution. We make use of this sample, as well as our current guess for the state action value at (s_{t+1}, a_{t+1}) to update the current guess for state action value at (s_t, a_t).

The exploration can be any general exploration policy like $\epsilon-$greedy or softmax. The role of the exploration policy is to enable enough exploration so as to move closer to the optimal policy while maintaining enough of exploitation. Convergence guarantees exist for $\epsilon-$greedy and $\epsilon-$soft policies([1]).

4.2 SARSA Control Algorithm

The SARSA control algorithm, in its entirety, is presented in Figure 5.

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Figure 5: SARSA Control Algorithm([1])

It is to be noted that SARSA usually estimates the approximately correct path to the target and not exactly the optimal path. Its aim is to stay as close to the optimal path as possible while not taking too many risks.

5 Q-Learning

Q-Learning is a **Temporal Difference** (TD) control algorithm for optimal policy learning. The key idea is to learn the state action value $q_\pi(s, a)$ for all the state action pairs for a policy π. However the difference from SARSA is that it directly estimates the optimal policy instead of simply updating the current policy to make it closer to the optimal policy.

Unlike SARSA, Q-Learning is an **Off-policy control algorithm**, in the sense that the updates to the action values are not made on the basis of the current policy but instead based on the optimal action performed at that state for the current set of Q-values. The Q-Learning control algorithm has two components: an update rule and an exploration policy.

5.1 Q-Learning Update Rule

The Q-Learning update rule is given as follows:

$$\boxed{Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot [r_t + \gamma \cdot \max_a(Q(s_{t+1}, a)) - Q(s_t, a_t)]}$$

The update rule can be interpreted as follows: The state action value, which represents the expected return, is updated each time we obtain a reward. Each reward can be thought of as a sample from the reward distribution. We make use of this sample, as well as the maximum of the state action values for all actions at that state, to update the current guess for state action value at (s_t, a_t).

The exploration can be any general exploration policy like $\epsilon-$greedy or softmax. The role of the exploration policy is to enable enough exploration so as to move closer to the optimal policy while maintaining enough of exploitation. Convergence guarantees exist for $\epsilon-$greedy and $\epsilon-$soft policies([1]).

5.2 Q-Learning Control Algorithm

The Q-Learning control algorithm, in its entirety, is presented in Figure 6.

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Figure 6: Q-Learning Control Algorithm([1])

In contrast to SARSA, Q-Learning is a much more optimistic and bold learning policy. It aims to estimate the optimal policy at the risk of accumulating negatuve rewards in the process. It is often termed as a more aggressive counterpart of SARSA.

6 Experimental Setup

We run a total of 32 experiments, using SARSA and Q-Learning control algorithms. For each, we run 16 experiments using different combinations of the following 4 parameters:

Parameter	Config1		Config2	
	Value	Code	Value	Code
Start State	(0,4)	S1	(3,6)	S2
Wind	False	Clear	True	Windy
Transition Probability	1.00	Determ(inistic)	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps	Softmax	Smx

It is to be noted that that the bias probability (b) is taken as 0.5. Although this value isn't provided in the problem statement, we used this value since this is the default value provided in the starter code. We also fix the number of episodes in one experiment to 10000 based on empirical evidence.

6.1 Hyperparameter Tuning

Apart from varying these parameters, we also tune the value of the following hyperparameters:

- α - The learning rate
- γ - The discount factor
- ϵ/ β for $\epsilon-$greedy/ softmax respectively

6.2 Optimality Measures

The problem statement requires us to make use of the "best" hyperparameters for each configuration of the parameters. While there could be multiple notions of what the "best" set of hyperparameters is, we filter out the hyperparameters based on the following notions of optimality:

- **Asymptotic behaviour:** We first filter out the set of hyperparameters which can asymptotically give us the best rewards. To check the asymptotic behaviour, we first run 10000 episodes using the hyperparameter configuration under consideration. We then adopt a **greedy** policy over the estimated q-values and use the total reward of this run as the metric for comparison. Using a greedy policy is essential to nullify any random effects caused due to exploration. All the hyperparameter configurations which give the (same) maximum reward are selected for further filtration.
- **Rate of convergence:** Once optimal asymptotic behaviour has been established, the next logical quantity to optimize for is the speed at which this behaviour is reached. We use a regret-like quantity as a proxy for this. For each hyperparameter configuration which exhibits best asymptotic behaviour, we run five experiments for 10000 episodes each and average out the rewards obtained in each episode. We then compute a regret-like quantity making use of the asymptotic reward (calculated during analysis of asymptotic behaviour). The set which minimizes this quantity is declared as the best hyperparameter set.

Now that the notion of the best hyperparameter set has been established, the next challenge is to actually devise methods of recovering them in a particular setting. Because of the stochastic nature of the entire setup, analytically optimizing for the best set of hyperparameters is rather unwieldy. Hence, we resort to a grid search approach.

6.3 Grid Search

Due to the limited compute available, we are restricted in the size of of the grid search space. This makes it essentially to make intelligent guesses of the "approximate" space in which the hyperparameters lie. These are described below:

- **Learning rate(α)**: We search over the set {0.001, 0.01, 0.1}. Since we are running 10000 episodes per experiment, which is quite a large count, it is reasonable to assume that smaller values of α would work well.
- **Discount factor(γ)**: We search over the set {0.8, 0.9, 1.0}. Since we have a fixed terminal state, γ close to 1 should work fine. This also ensures that our model is not very myopic.
- **Epsilon (ϵ)**: We search over the set {0.001, 0.01, 0.1}. These values have been proven to work well empirically.
- **Temperature (β)**: We search over the set {0.5, 1.0, 5.0}. These values are chosen since the magnitude of the Q-values usually stays less than or equal to 10.

It is important to note that the above values constitute the coarse grid search done for all cases. In specific variants of the problem where these combinations of hyperparameters have failed to give good results, we have carried out fine tuning of the various hyperparameters based on the rewards obtained in each case. This is done on a case-by-case basis.

7 SARSA: Experimental Study

This section contains details of all experiments conducted for the various variants of the Grid-world problem using the SARSA update policy

7.1 Experiment 1: [S1, Clear, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 1: Parameters and Hyper-parameters: SARSA Exp-1

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

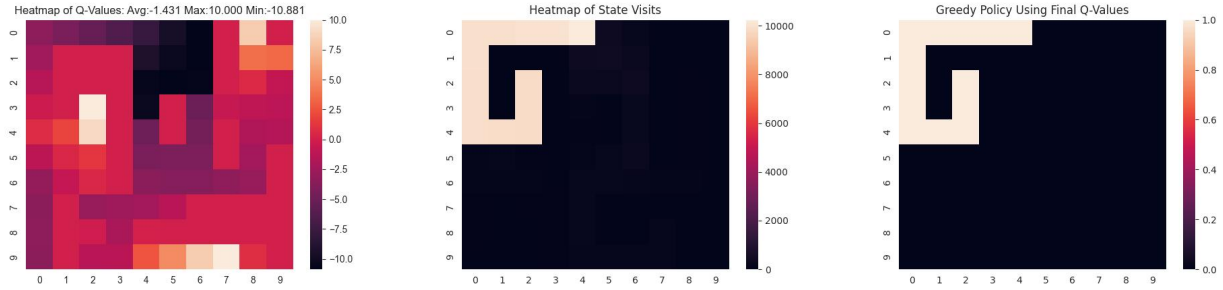


Figure 7: Heatmaps for SARSA Exp-1

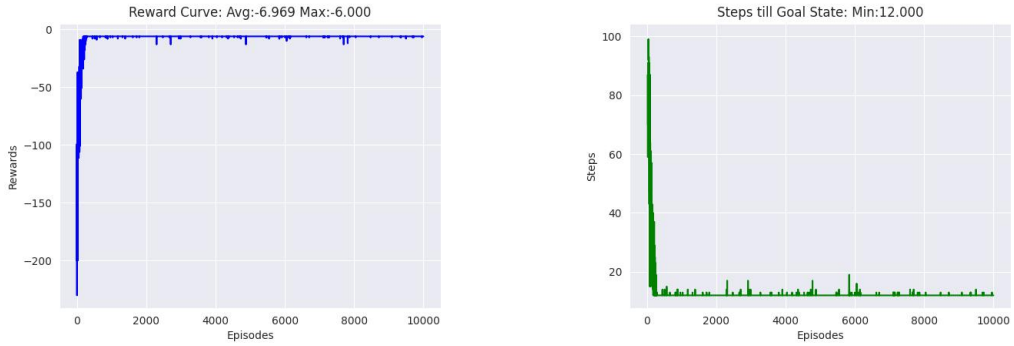


Figure 8: Plots for SARSA Exp-1

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

7.2 Experiment 2: [S1, Clear, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.5

(b) Best Hyperparameters

Table 2: Parameters and Hyper-parameters: SARSA Exp-2

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

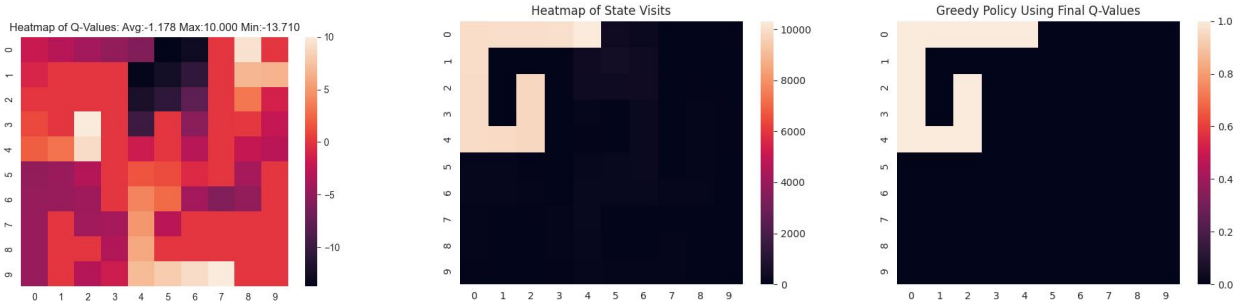


Figure 9: Heatmaps for SARSA Exp-2

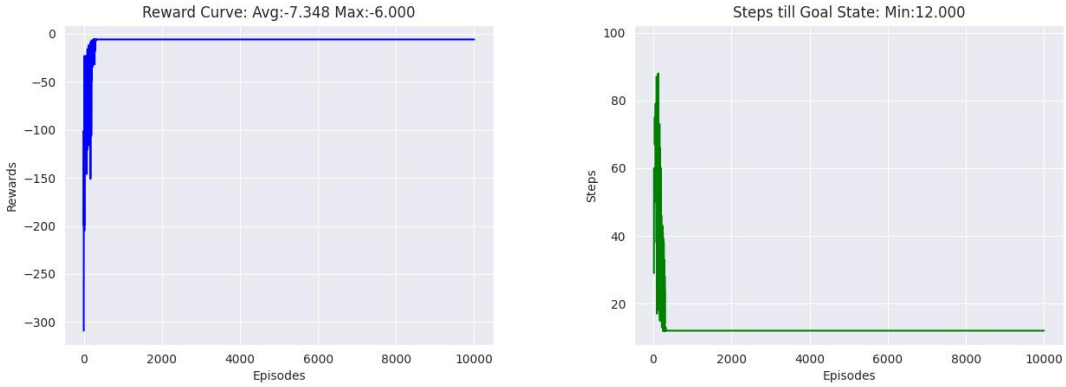


Figure 10: Plots for SARSA Exp-2

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much

more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax.

7.3 Experiment 3: [S1, Clear, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.07
γ - DF	0.90
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 3: Parameters and Hyper-parameters: SARSA Exp-3

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

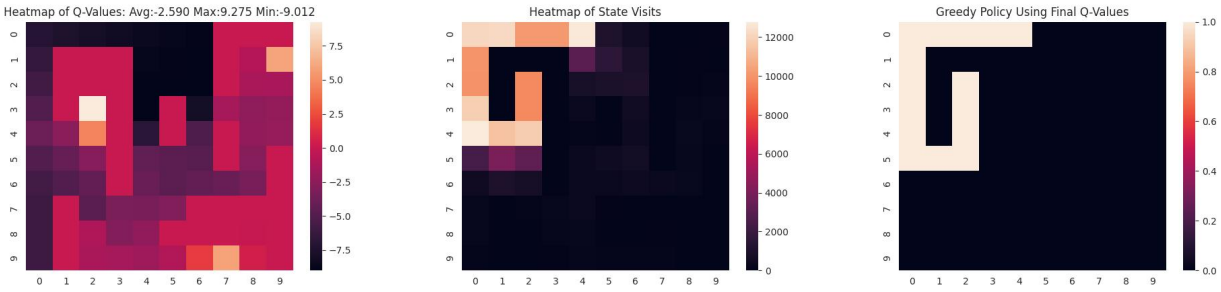


Figure 11: Heatmaps for SARSA Exp-3

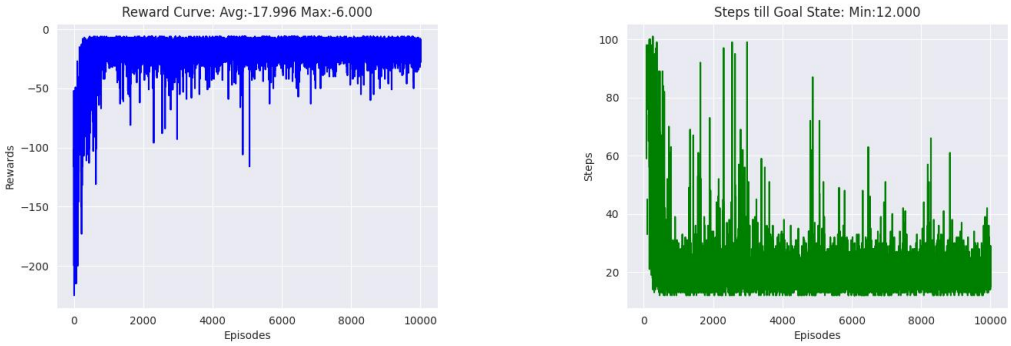


Figure 12: Plots for SARSA Exp-3

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

7.4 Experiment 4: [S1, Clear, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	0.90
β - temperature	0.05

(b) Best Hyperparameters

Table 4: Parameters and Hyper-parameters: SARSA Exp-4

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

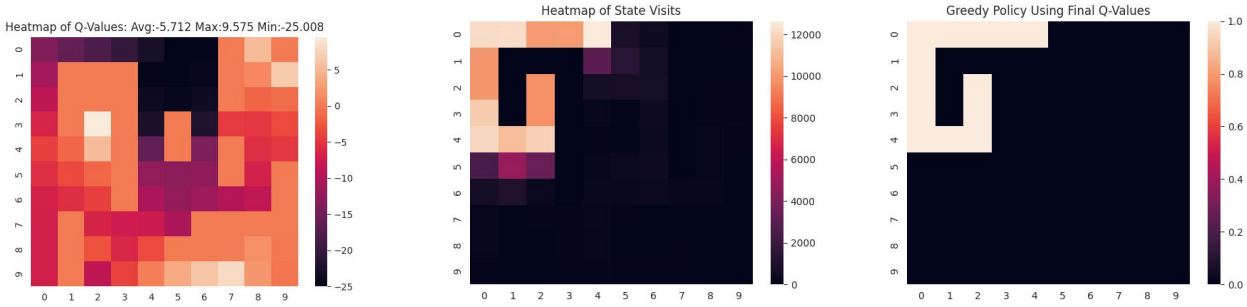


Figure 13: Heatmaps for SARSA Exp-4

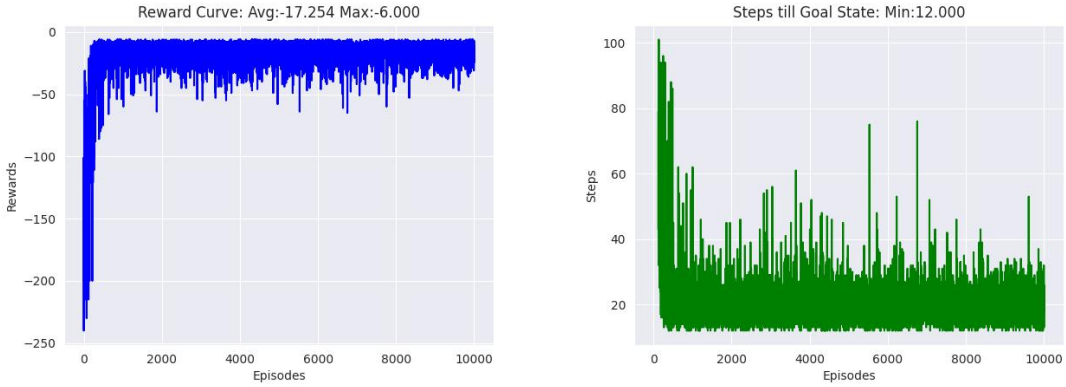


Figure 14: Plots for SARSA Exp-4

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit, as we saw above in the case of $\epsilon-$Greedy.

7.5 Experiment 5: [S1, Windy, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.01
γ - DF	1.00
ϵ (ϵ-greedy)	0.01

(b) Best Hyperparameters

Table 5: Parameters and Hyper-parameters: SARSA Exp-5

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

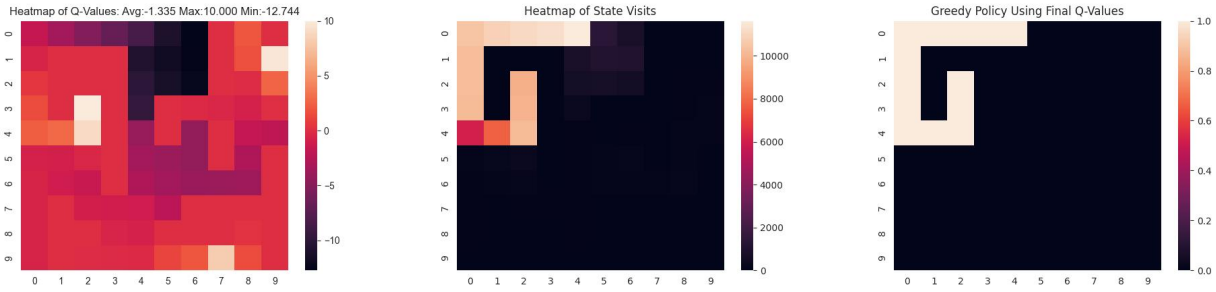


Figure 15: Heatmaps for SARSA Exp-5

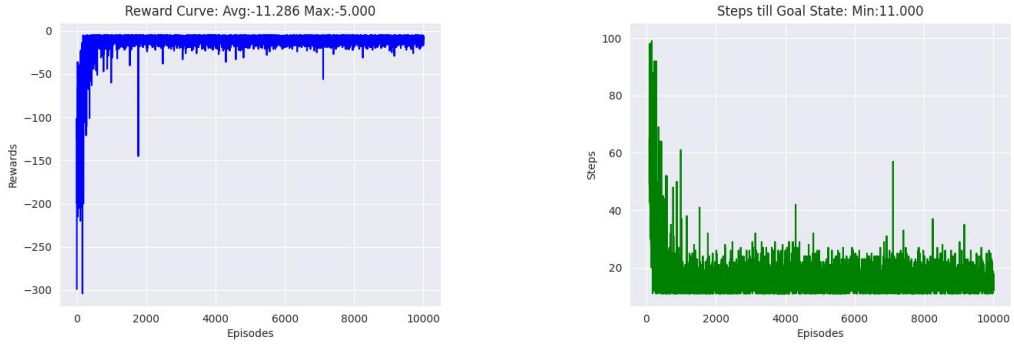


Figure 16: Plots for SARSA Exp-5

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

7.6 Experiment 6: [S1, Windy, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.10

(b) Best Hyperparameters

Table 6: Parameters and Hyper-parameters: SARSA Exp-6

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

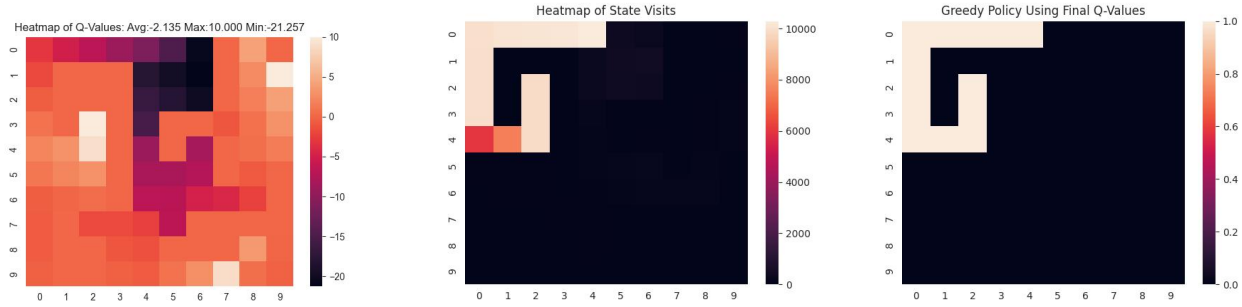


Figure 17: Heatmaps for SARSA Exp-6

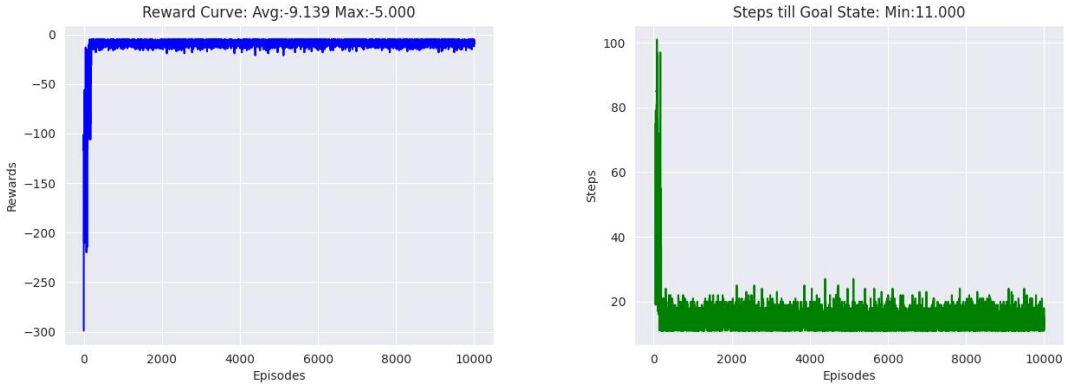


Figure 18: Plots for SARSA Exp-6

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax.

7.7 Experiment 7: [S1, Windy, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.05
γ - DF	0.90
ϵ (ϵ-greedy)	0.05

(b) Best Hyperparameters

Table 7: Parameters and Hyper-parameters: SARSA Exp-7

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

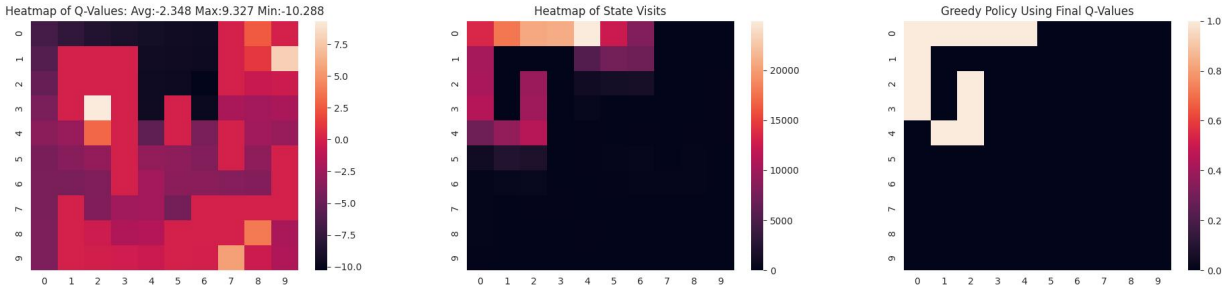


Figure 19: Heatmaps for SARSA Exp-7

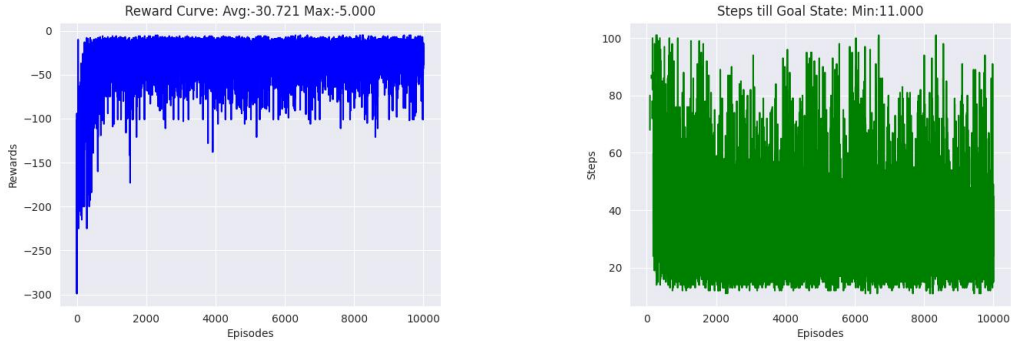


Figure 20: Plots for SARSA Exp-7

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

7.8 Experiment 8: [S1, Windy, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.05

(b) Best Hyperparameters

Table 8: Parameters and Hyper-parameters: SARSA Exp-8

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

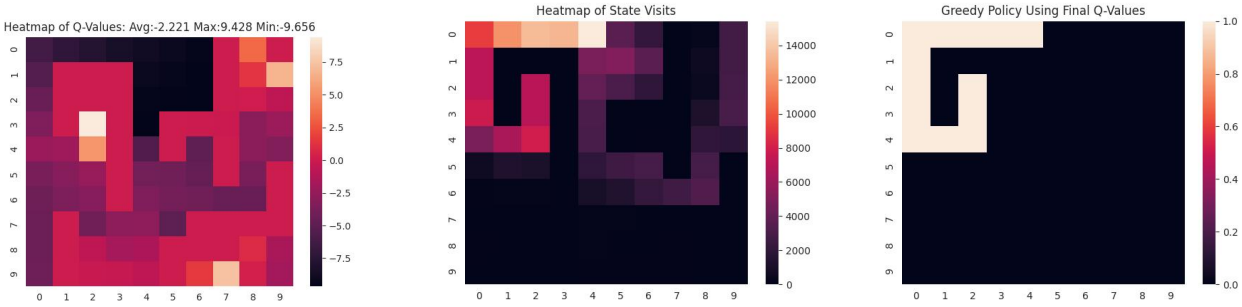


Figure 21: Heatmaps for SARSA Exp-8

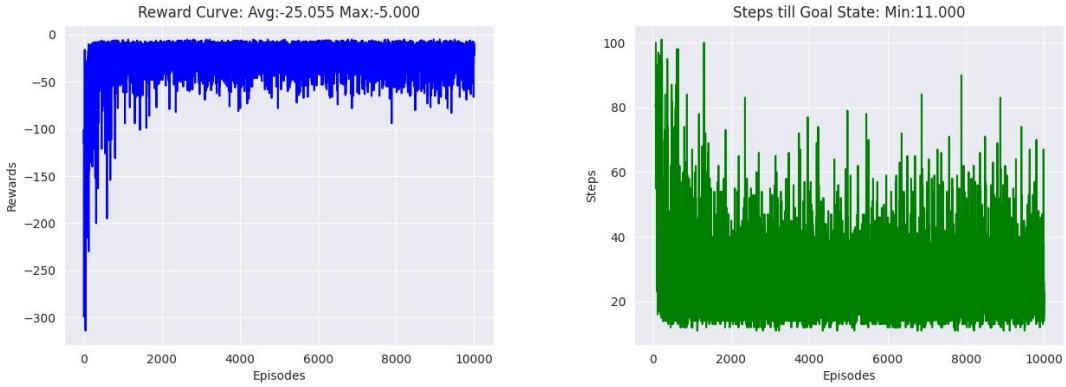


Figure 22: Plots for SARSA Exp-8

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

7.9 Experiment 9: [S2, Clear, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 9: Parameters and Hyper-parameters: SARSA Exp-9

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

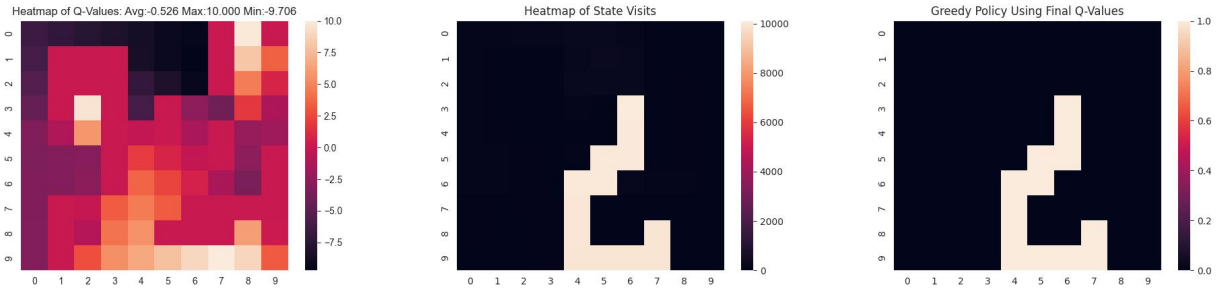


Figure 23: Heatmaps for SARSA Exp-9

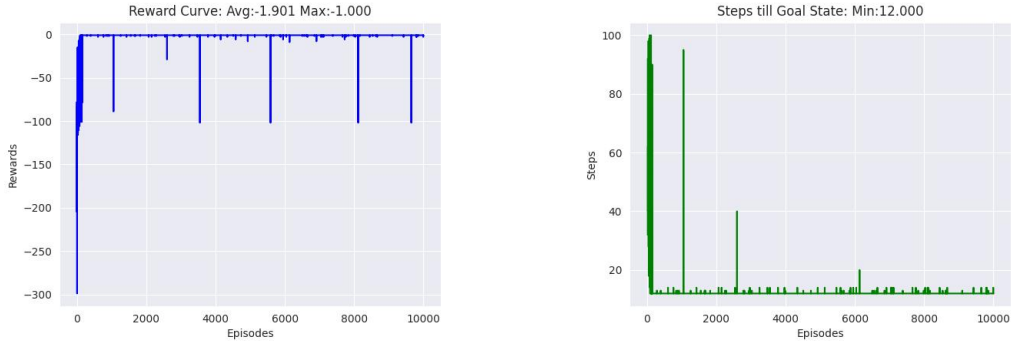


Figure 24: Plots for SARSA Exp-9

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

7.10 Experiment 10: [S2, Clear, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.5

(b) Best Hyperparameters

Table 10: Parameters and Hyper-parameters: SARSA Exp-10

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

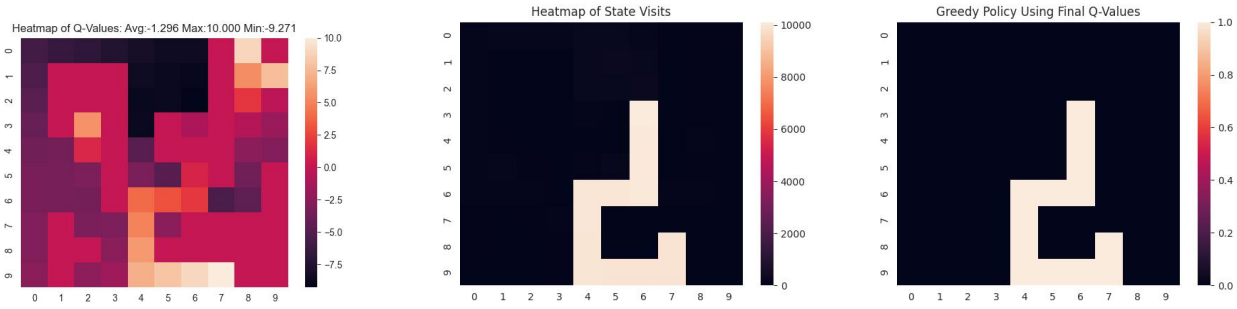


Figure 25: Heatmaps for SARSA Exp-10

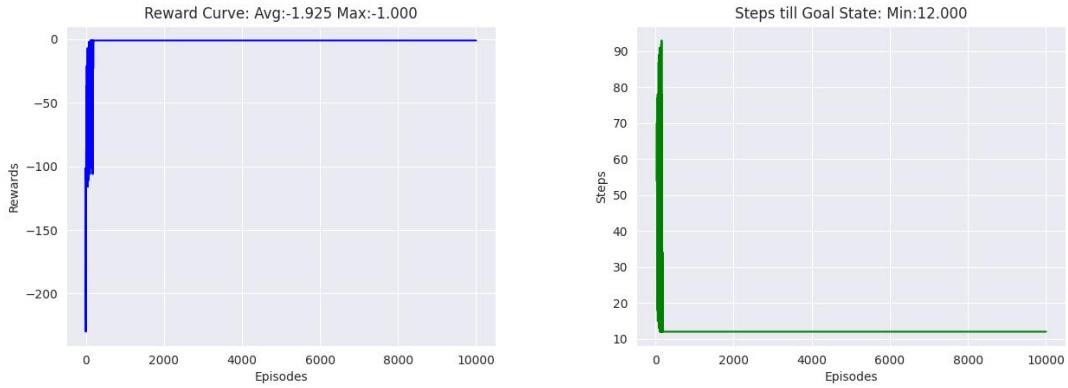


Figure 26: Plots for SARSA Exp-10

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax.

7.11 Experiment 11: [S2, Clear, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	0.90
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 11: Parameters and Hyper-parameters: SARSA Exp-11

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

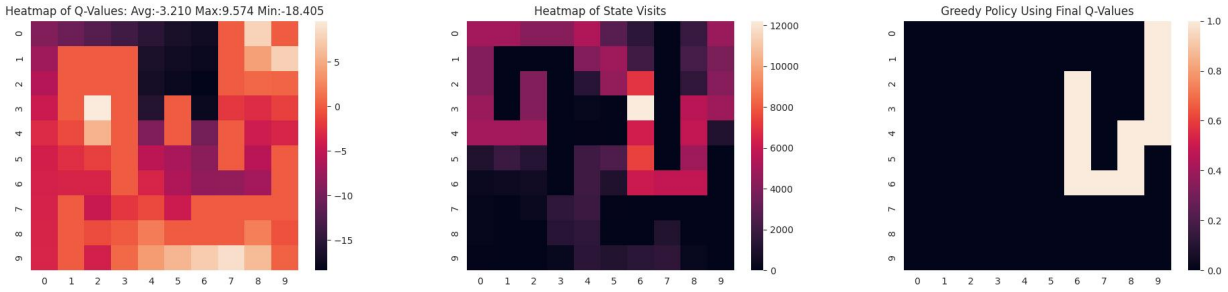


Figure 27: Heatmaps for SARSA Exp-11

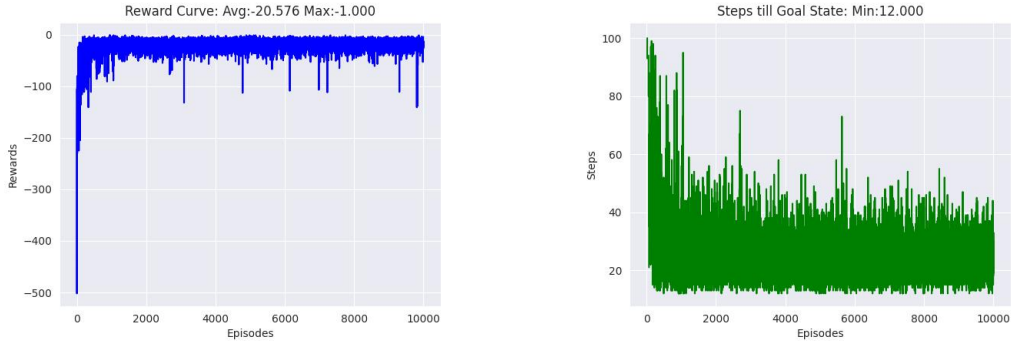


Figure 28: Plots for SARSA Exp-11

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

7.12 Experiment 12: [S2, Clear, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.05
γ - DF	1.00
β - temperature	0.05

(b) Best Hyperparameters

Table 12: Parameters and Hyper-parameters: SARSA Exp-12

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

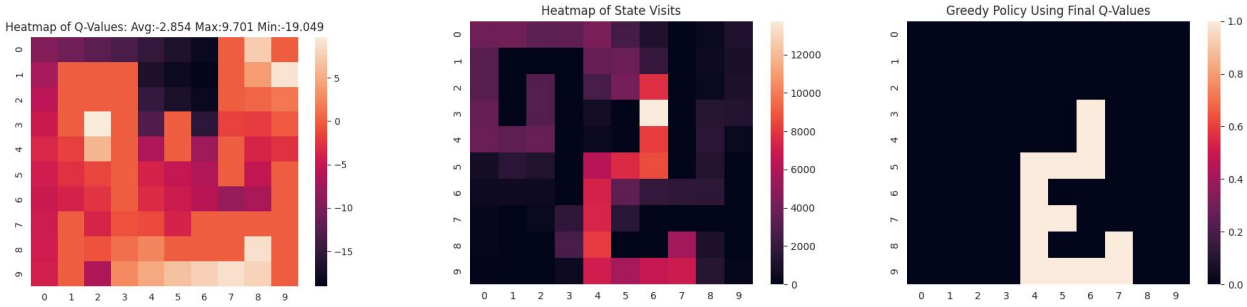


Figure 29: Heatmaps for SARSA Exp-12

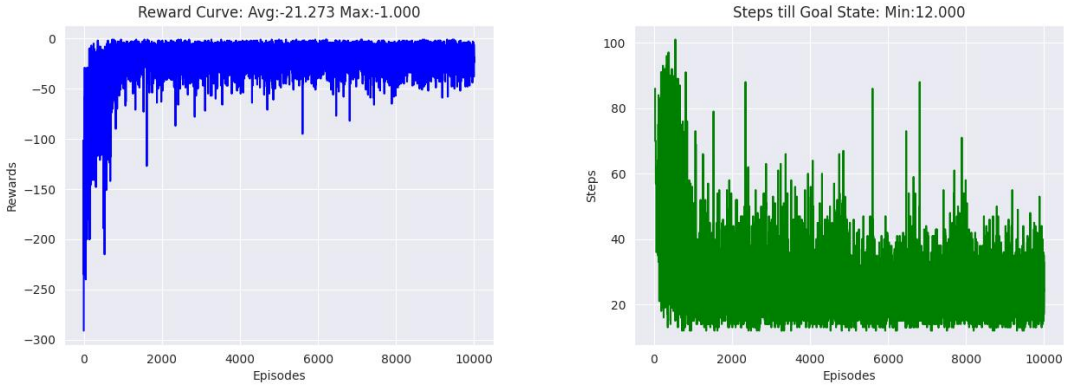


Figure 30: Plots for SARSA Exp-12

From the above plots for reward and steps, it is quite clear that the case is similar to Experiment 11. The noisy nature of actions make it very difficult for the agent to learn.

7.13 Experiment 13: [S2, Windy, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.1
γ - DF	1.00
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 13: Parameters and Hyper-parameters: SARSA Exp-13

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

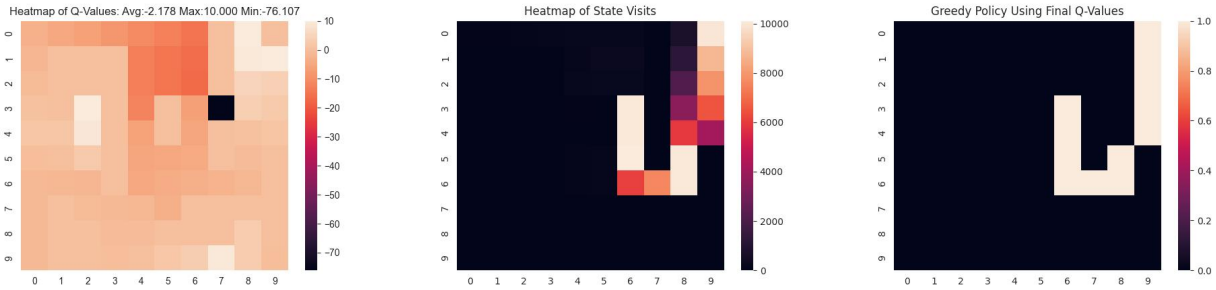


Figure 31: Heatmaps for SARSA Exp-13

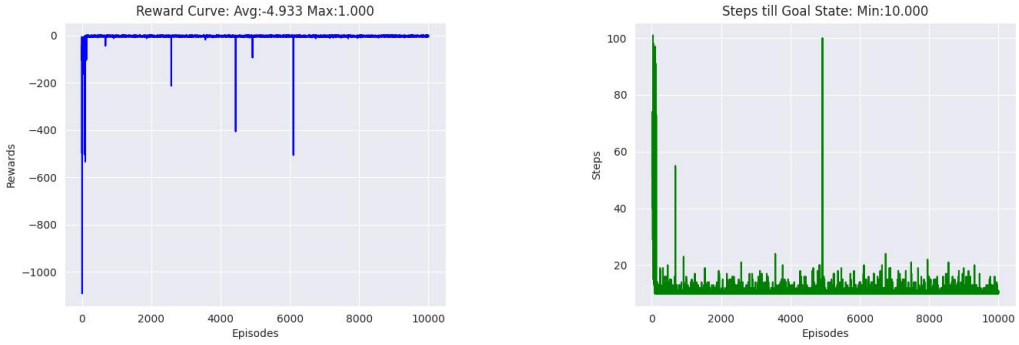


Figure 32: Plots for SARSA Exp-13

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

7.14 Experiment 14: [S2, Windy, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.05

(b) Best Hyperparameters

Table 14: Parameters and Hyper-parameters: SARSA Exp-14

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

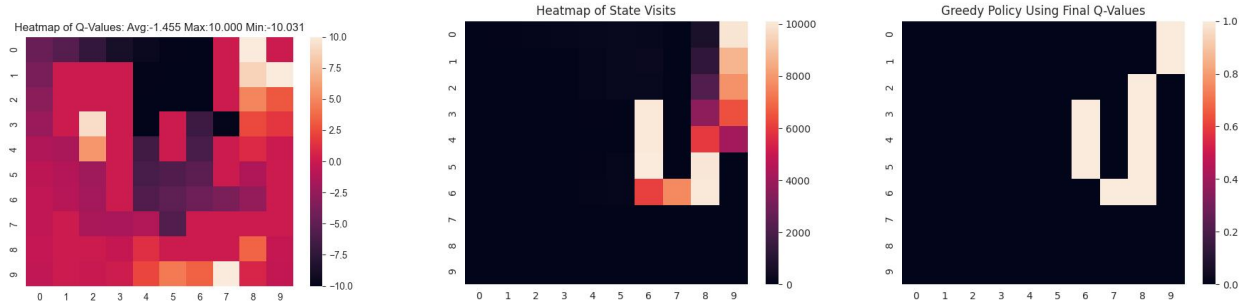


Figure 33: Heatmaps for SARSA Exp-14

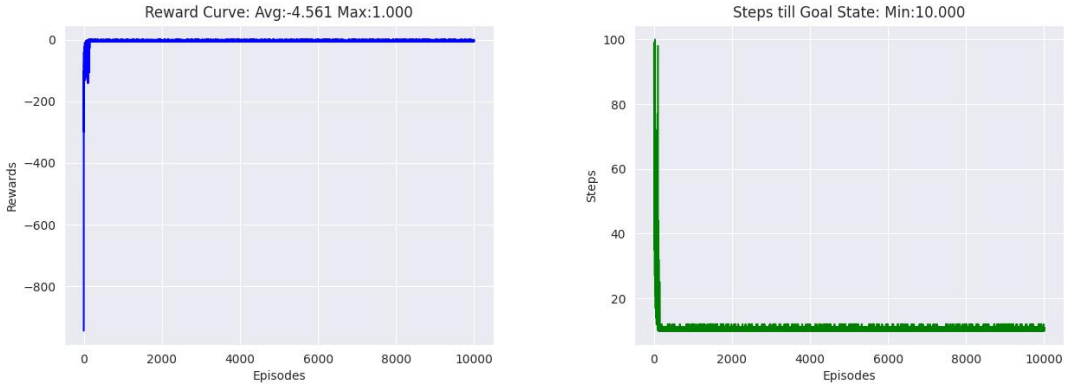


Figure 34: Plots for SARSA Exp-14

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax.

7.15 Experiment 15: [S2, Windy, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.03
γ - DF	0.90
ϵ (ϵ-greedy)	0.005

(b) Best Hyperparameters

Table 15: Parameters and Hyper-parameters: SARSA Exp-15

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

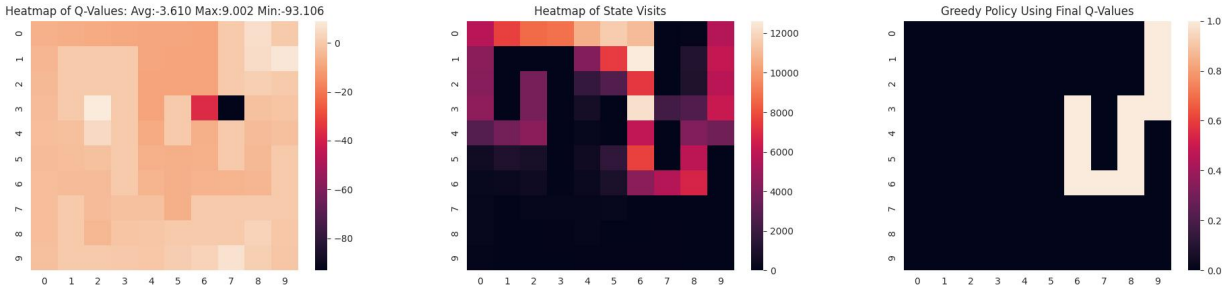


Figure 35: Heatmaps for SARSA Exp-15

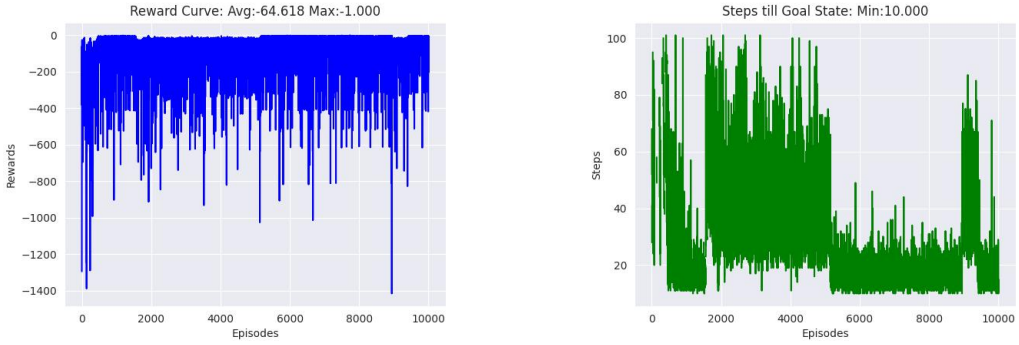


Figure 36: Plots for SARSA Exp-15

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

7.16 Experiment 16: [S2, Windy, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.005
γ - DF	0.80
β - temperature	0.05

(b) Best Hyperparameters

Table 16: Parameters and Hyper-parameters: SARSA Exp-16

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

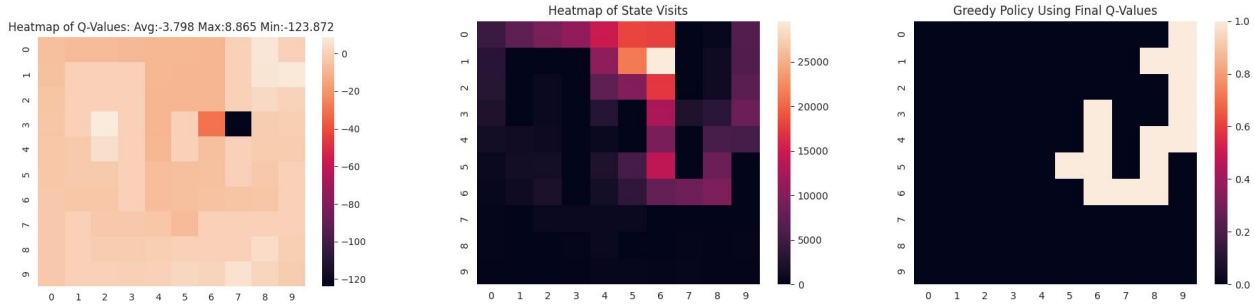


Figure 37: Heatmaps for SARSA Exp-16

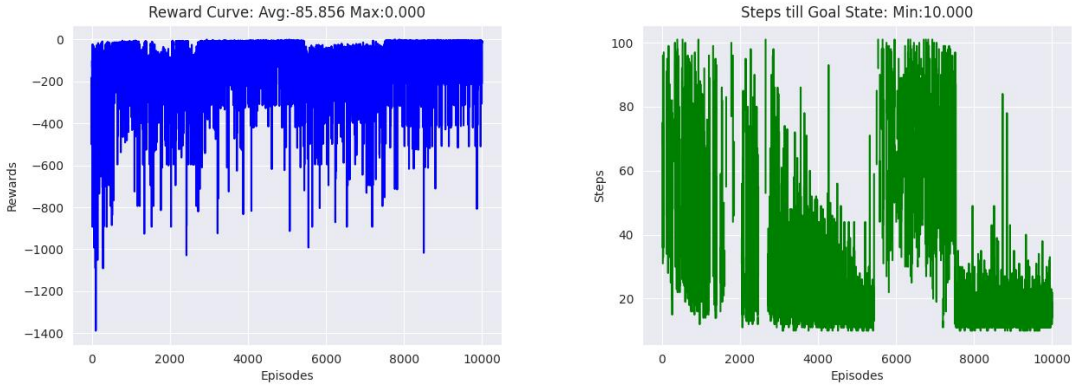


Figure 38: Plots for SARSA Exp-16

From the above plots for reward and steps, it is quite clear that the case is similar to Experiment 15. The noisy nature of actions make it very difficult for the agent to learn.

8 Q-Learning: Experimental Study

This section contains details of all experiments conducted for the various variants of the Grid-world problem using the Q-Learning update policy.

8.1 Experiment 1: [S1, Clear, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 17: Parameters and Hyper-parameters: QLearning Exp-1

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

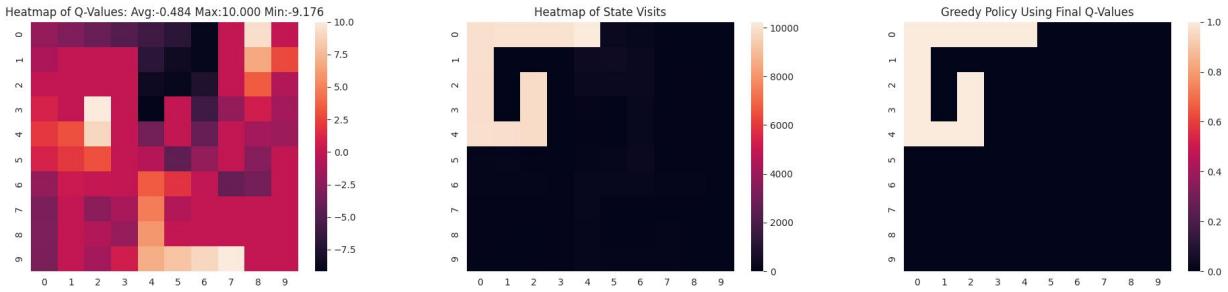


Figure 39: Heatmaps for QLearning Exp-1

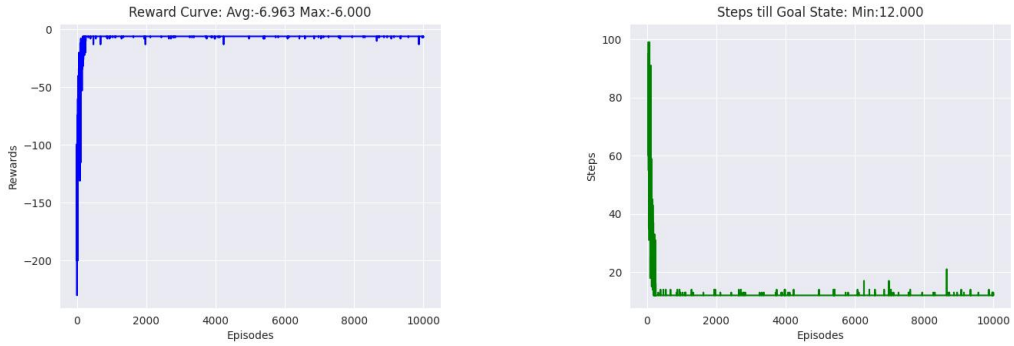


Figure 40: Plots for QLearning Exp-1

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state. Both SARSA and QLearning perform similarly here.

8.2 Experiment 2: [S1, Clear, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
β - temperature	0.5

(b) Best Hyperparameters

Table 18: Parameters and Hyper-parameters: QLearning Exp-2

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

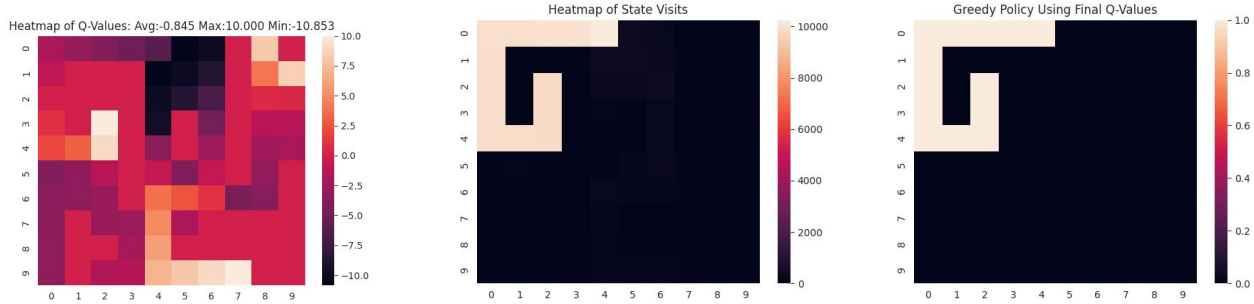


Figure 41: Heatmaps for QLearning Exp-2

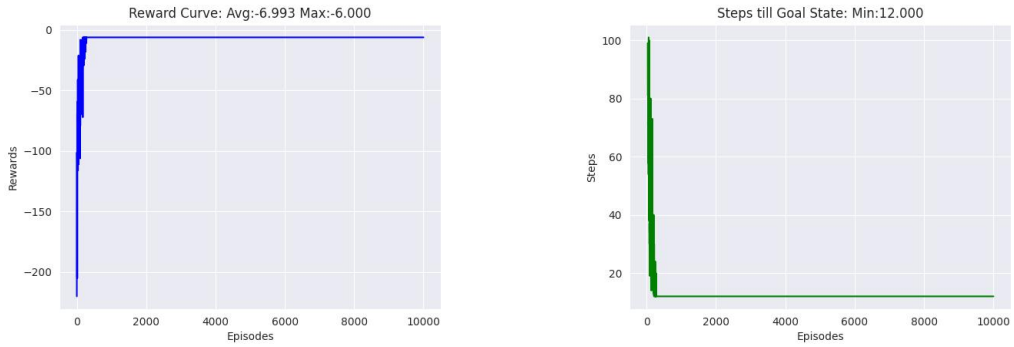


Figure 42: Plots for QLearning Exp-2

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax. Both SARSA and QLearning perform similarly here.

8.3 Experiment 3: [S1, Clear, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.03
γ - DF	0.90
ϵ (ϵ-greedy)	0.01

(b) Best Hyperparameters

Table 19: Parameters and Hyper-parameters: QLearning Exp-3

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

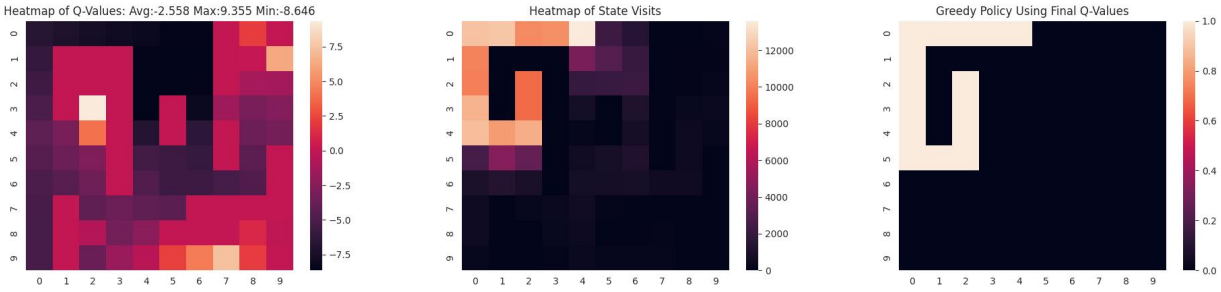


Figure 43: Heatmaps for QLearning Exp-3

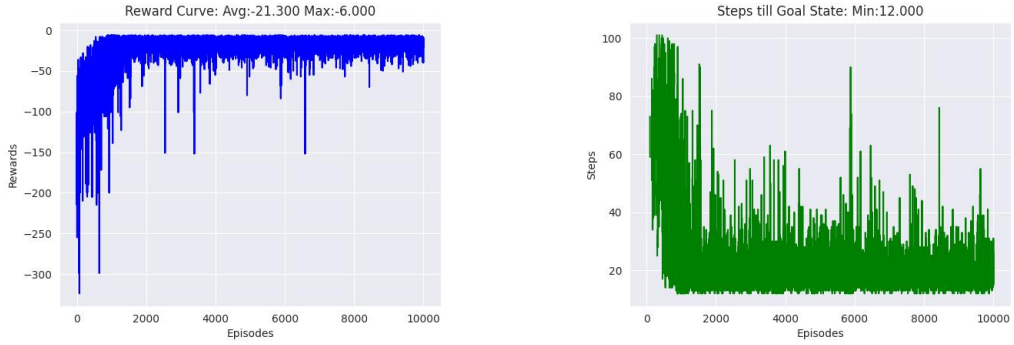


Figure 44: Plots for QLearning Exp-3

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy. It can be seen that in the plot of learnt policy, the agent moves to some unnecessary cells. This is due to the lack of control the agent has over its own actions due to the inherent stochasticity/ noise.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficult to lower the number of steps taken beyond a certain limit.

8.4 Experiment 4: [S1, Clear, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.05
γ - DF	0.90
β - temperature	0.0005

(b) Best Hyperparameters

Table 20: Parameters and Hyper-parameters: QLearning Exp-4

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

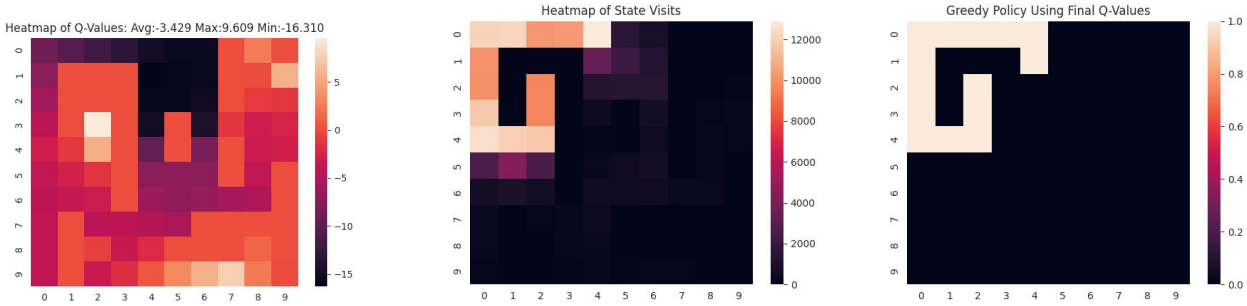


Figure 45: Heatmaps for QLearning Exp-4

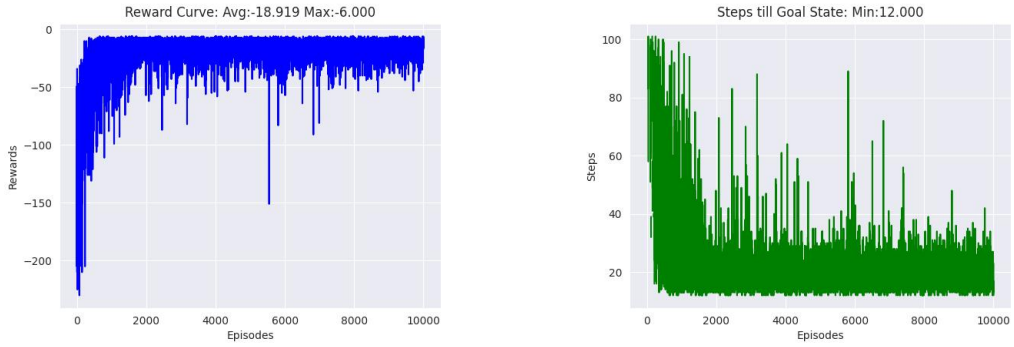


Figure 46: Plots for QLearning Exp-4

From the above plots for reward and steps, we can see that the agent does not perform great. Moreover an interesting point here is that based on the heatmap of state visits we would expect the agent to choose a path to the goal state at $(2, 2)$. However it is the stochasticity in action control that causes the agent to make the first step downward, instead of leftward. From this new point it is easier to take the step towards either of the two other goal states which it does.

It is important to note that the optimal path towards either of these goal states takes the agent close to a restart state, which is the reason for many of the rewards especially during initial timesteps to have otherwise unexplained penalties of over -100.

8.5 Experiment 5: [S1, Windy, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	1.00
ϵ (ϵ-greedy)	0.10

(b) Best Hyperparameters

Table 21: Parameters and Hyper-parameters: QLearning Exp-5

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

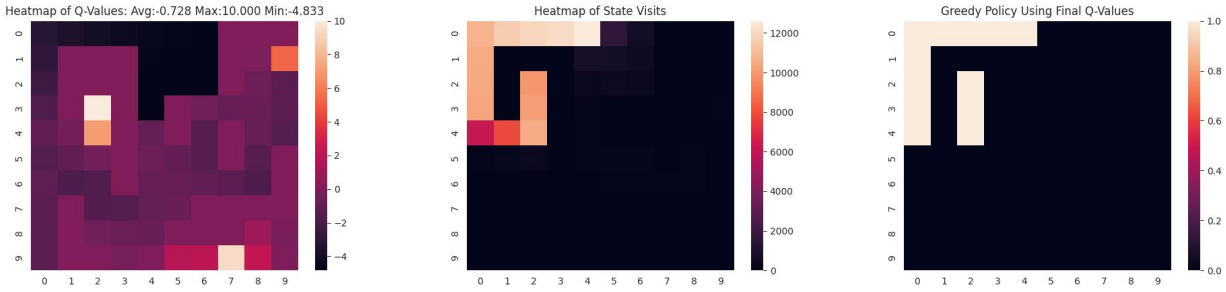


Figure 47: Heatmaps for QLearning Exp-5

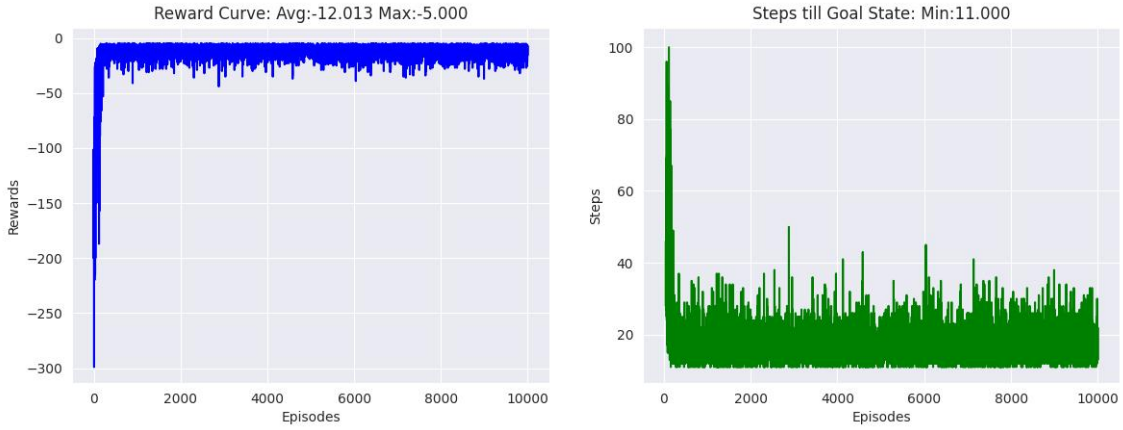


Figure 48: Plots for QLearning Exp-5

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state. Both SARSA and QLearning perform similarly here.

8.6 Experiment 6: [S1, Windy, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	0.90
β - temperature	0.005

(b) Best Hyperparameters

Table 22: Parameters and Hyper-parameters: QLearning Exp-6

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

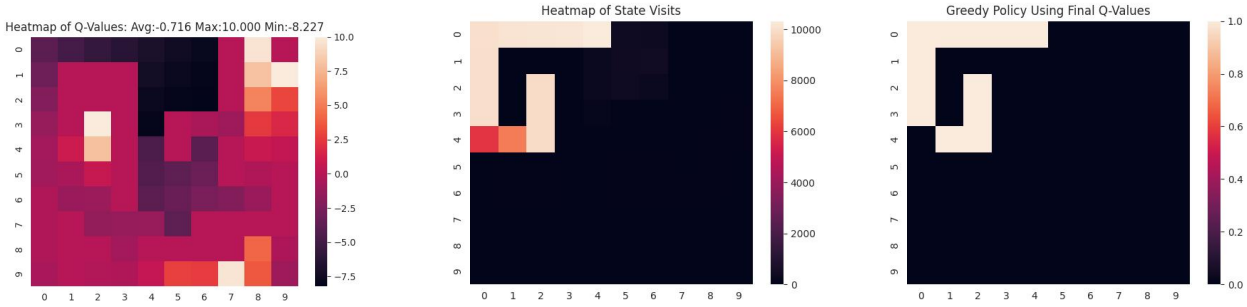


Figure 49: Heatmaps for QLearning Exp-6

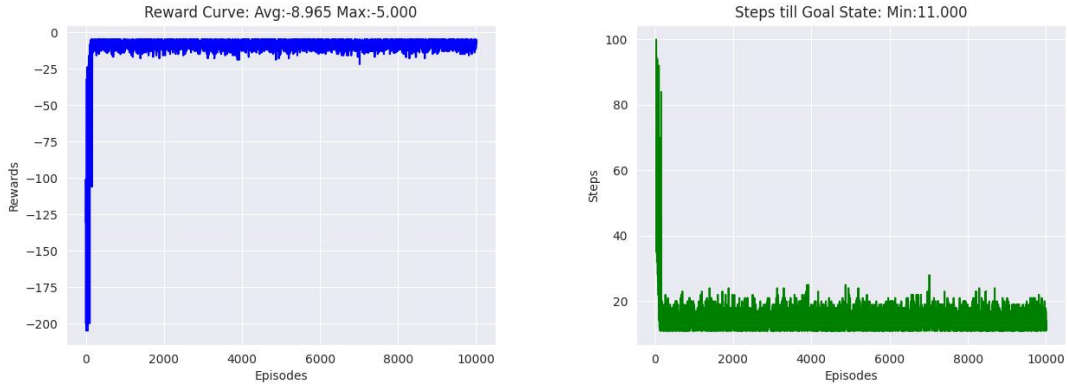


Figure 50: Plots for QLearning Exp-6

We observe that despite there being the presence of wind, the agent learns a good policy. This could be due to the deterministic direction of wind, i.e., even though the wind itself is stochastic, since the direction is fixed, the agent is able to adapt to it.

8.7 Experiment 7: [S1, Windy, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.03
γ - DF	1.00
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 23: Parameters and Hyper-parameters: QLearning Exp-7

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

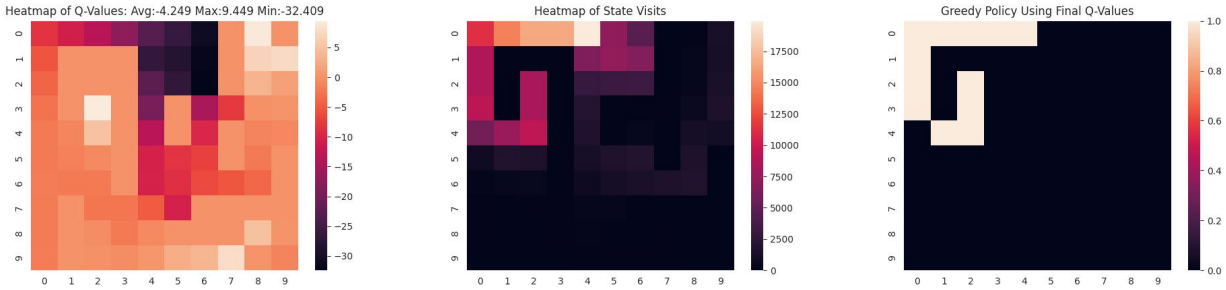


Figure 51: Heatmaps for QLearning Exp-7

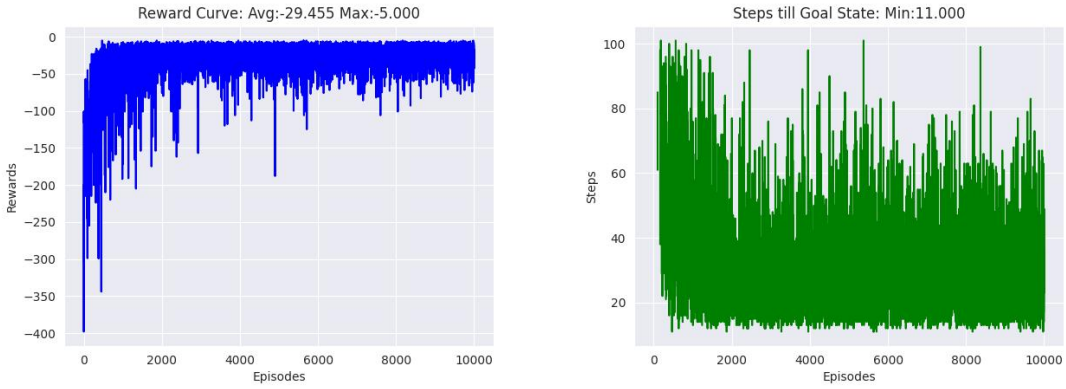


Figure 52: Plots for QLearning Exp-7

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

8.8 Experiment 8: [S1, Windy, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(0,4)	S1
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.005
γ - DF	1.0
β - temperature	0.01

(b) Best Hyperparameters

Table 24: Parameters and Hyper-parameters: QLearning Exp-8

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

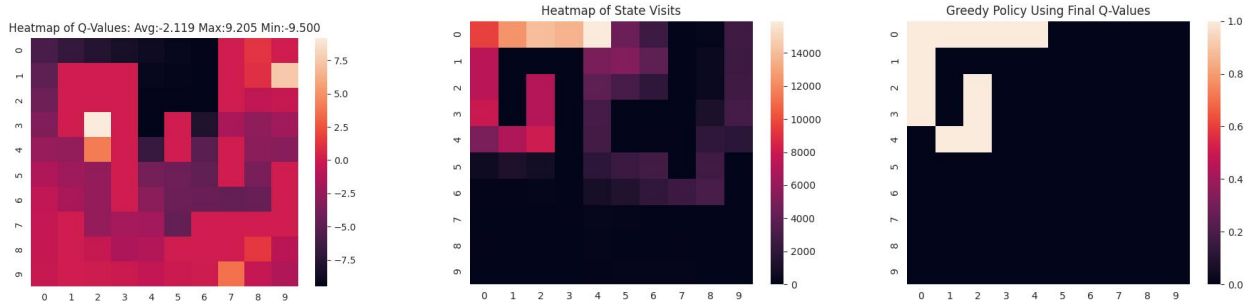


Figure 53: Heatmaps for QLearning Exp-8

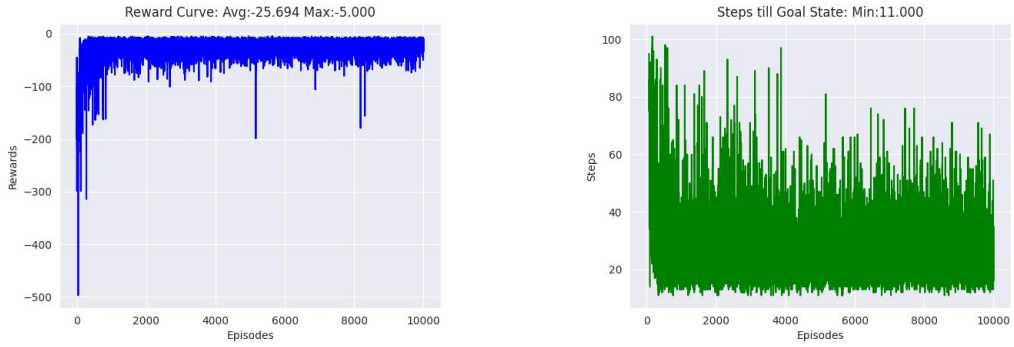


Figure 54: Plots for QLearning Exp-8

Despite stochasticity being introduced both in actions as well as wind playing a role, the agent is able to arrive at an optimal policy at the end. However, the number of steps as well as regret is rather high due to the highly stochastic environment.

8.9 Experiment 9: [S2, Clear, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.1
γ - DF	1.0
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 25: Parameters and Hyper-parameters: QLearning Exp-9

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

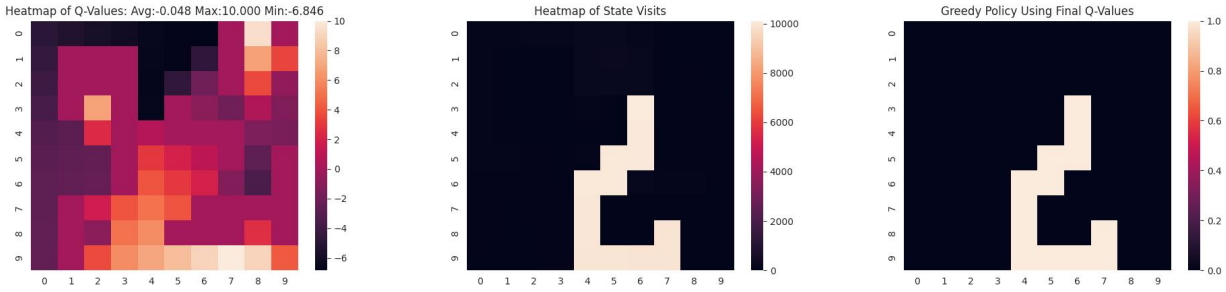


Figure 55: Heatmaps for QLearning Exp-9

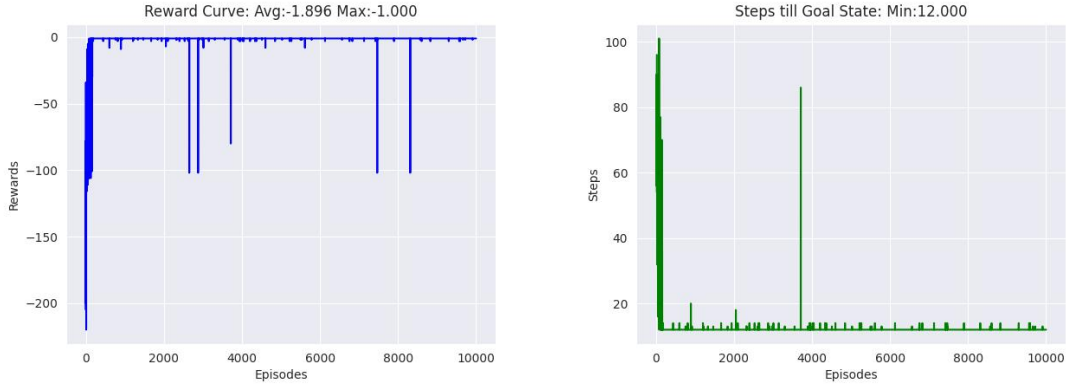


Figure 56: Plots for QLearning Exp-9

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S1 to the nearest goal state.

8.10 Experiment 10: [S2, Clear, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.1
γ - DF	1.0
β - temperature	0.05

(b) Best Hyperparameters

Table 26: Parameters and Hyper-parameters: QLearning Exp-10

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

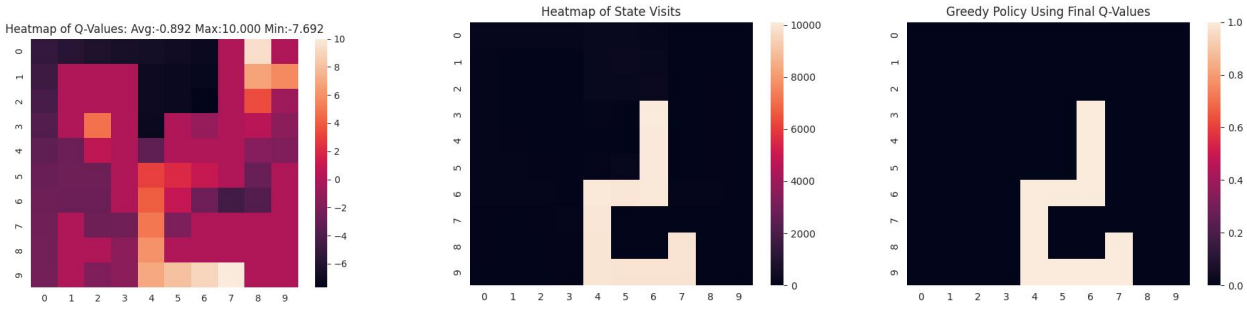


Figure 57: Heatmaps for QLearning Exp-10

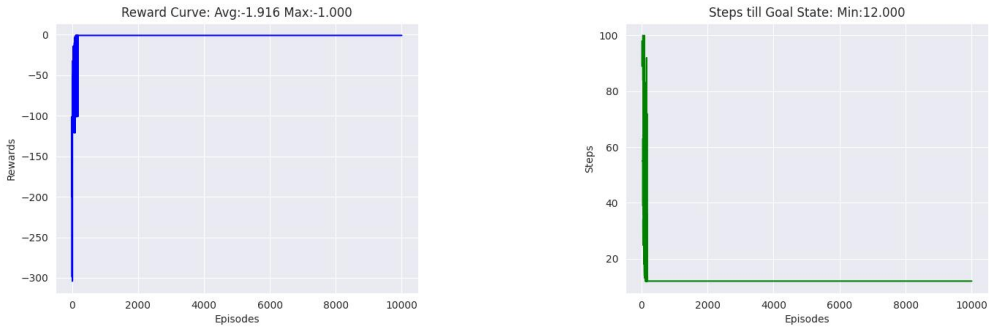


Figure 58: Plots for QLearning Exp-10

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns the optimal path from the start state S2 to the nearest goal state. The heatmaps for state visits agrees with this as does the Q-value heatmap which shows that the agent visited all three goal states during its exploration and learnt the nearest state.

An important difference to be noted when compared to ϵ-greedy is that once the optimal path is obtained the Q-values of states along the path are higher and the ratio of probabilities heavily favours this path much more than in the case of ϵ-greedy because of how softmax is calculated. Ratio of exploration is quickly brought down by softmax, hence its much better average reward. Both SARSA and QLearning perform similarly here.

8.11 Experiment 11: [S2, Clear, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.05
γ - DF	1.00
ϵ (ϵ-greedy)	0.01

(b) Best Hyperparameters

Table 27: Parameters and Hyper-parameters: QLearning Exp-11

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

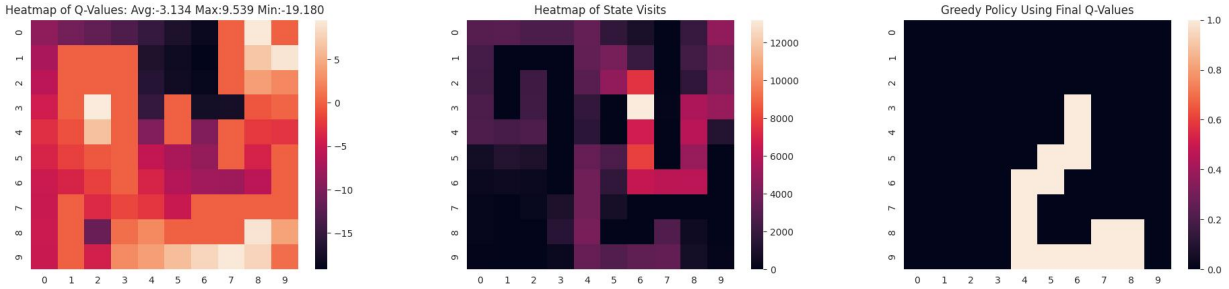


Figure 59: Heatmaps for QLearning Exp-11

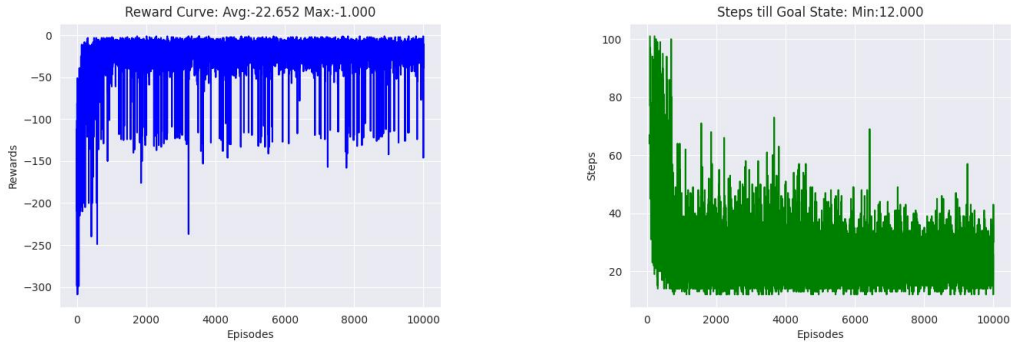


Figure 60: Plots for QLearning Exp-11

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

Furthermore the start state is right next to a restart state. Thus there is a 30% chance that the first move leads to a -100 reward regardless of what action is chosen (up and down are the only possible movements here). This explains the abnormally high penalties which do not change well into training.

8.12 Experiment 12: [S2, Clear, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	False	Clear
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.05
γ - DF	0.90
β - temperature	0.0005

(b) Best Hyperparameters

Table 28: Parameters and Hyper-parameters: QLearning Exp-12

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

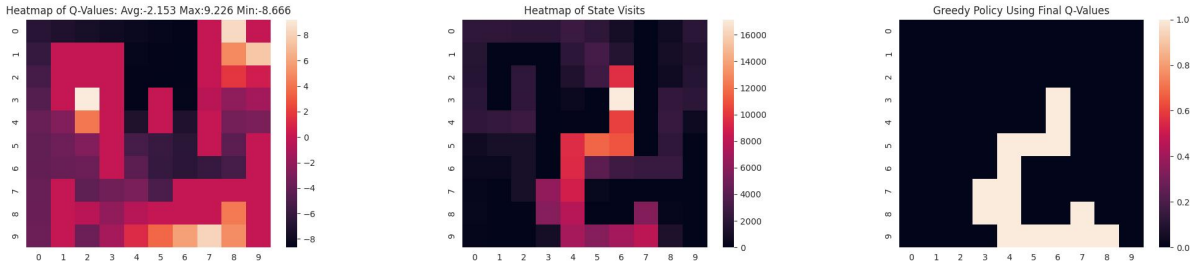


Figure 61: Heatmaps for QLearning Exp-12

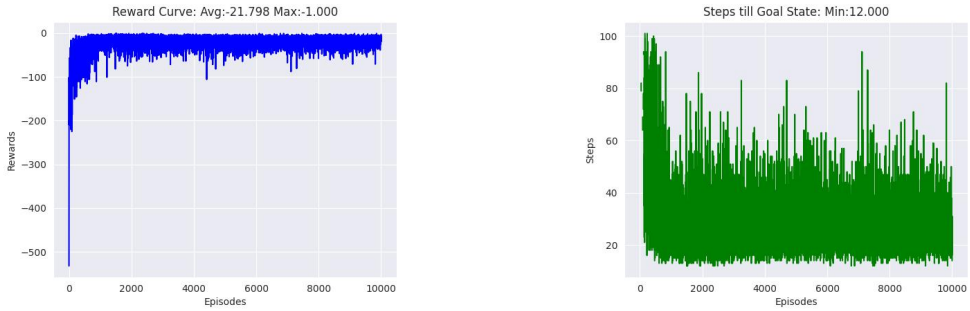


Figure 62: Plots for QLearning Exp-12

From the above plots for reward and steps, we can see that the agent does not perform great. The presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

Once again it is important to note that the start state is right next to a restart state. Thus there is a 30% chance that the first move leads to a -100 reward regardless of what action is chosen (up and down are the only possible movements here). This explains the abnormally high penalties which do not change well into training.

Again softmax performs poorer than ϵ-greedy because the probabilities are controlled by the action values. When actions get penalized the chances of them being taken progressively decrease in softmax. The heavy penalty of -100 thus throws the agent off track.

8.13 Experiment 13: [S2, Windy, Determ, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.10
γ - DF	0.90
ϵ (ϵ-greedy)	0.001

(b) Best Hyperparameters

Table 29: Parameters and Hyper-parameters: QLearning Exp-13

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

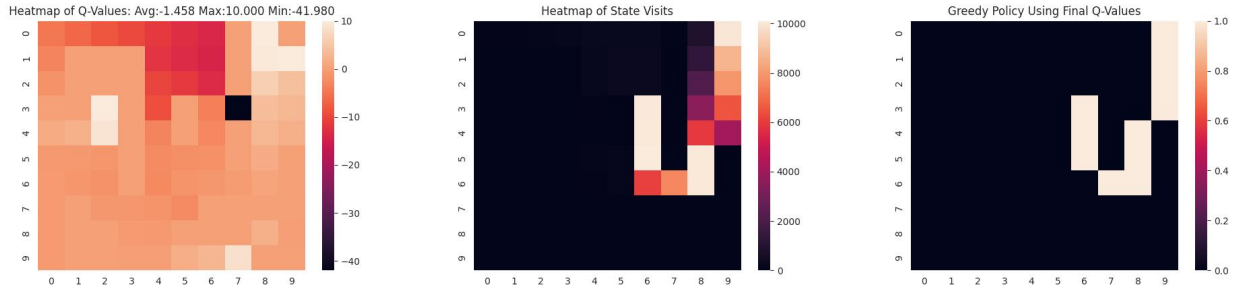


Figure 63: Heatmaps for QLearning Exp-13

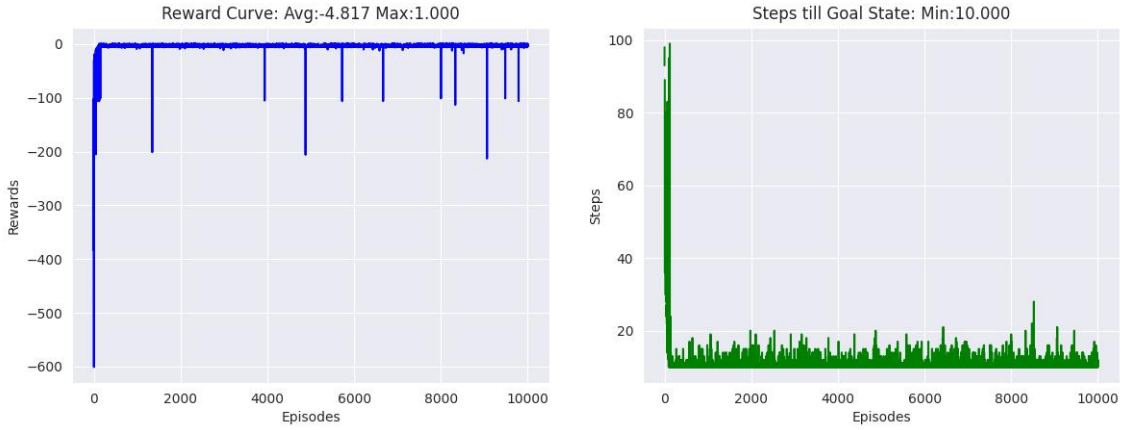


Figure 64: Plots for QLearning Exp-13

From the above plots for reward and steps, it is clear that that the RL agent performs very well and learns a more-or-less the optimal path from the start state S2 to the nearest goal state.

We can see the effect of the wind in the last leg of the agent's journey where the wind pushes it to the right and causes it to pass through the bad state incurring extra penalty. The heatmap also tells us a similar story.

8.14 Experiment 14: [S2, Windy, Determ, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	1.00	Determ
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.1
γ - DF	0.9
β - temperature	0.005

(b) Best Hyperparameters

Table 30: Parameters and Hyper-parameters: QLearning Exp-14

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

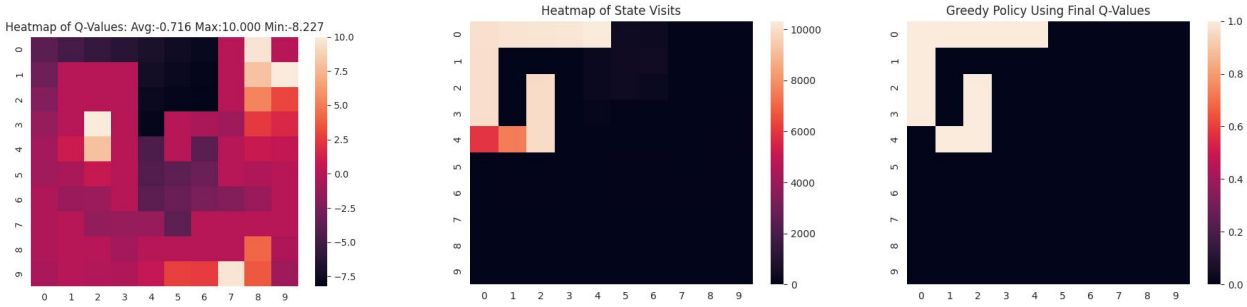


Figure 65: Heatmaps for QLearning Exp-14

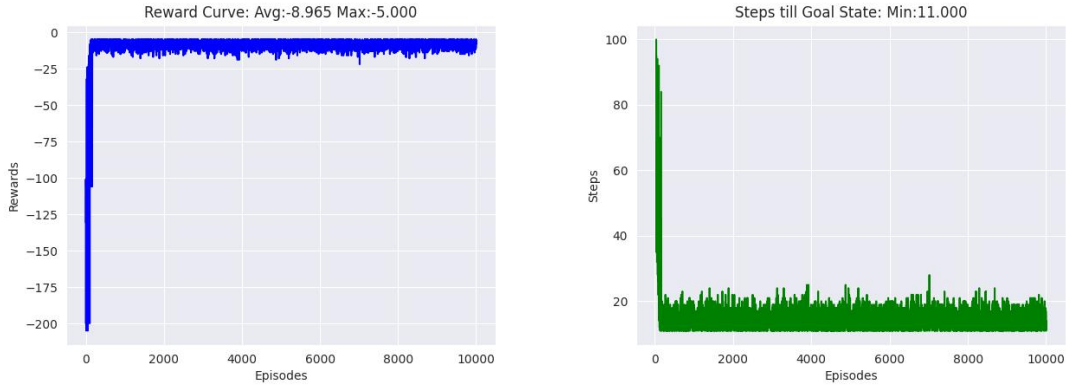


Figure 66: Plots for QLearning Exp-14

We observe that despite there being the presence of wind, the agent learns a good policy. This could be due to the deterministic direction of wind, i.e., even though the wind itself is stochastic, since the direction is fixed, the agent is able to adapt to it.

8.15 Experiment 15: [S2, Windy, Noisy, Eps]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	ϵ-Greedy	Eps

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.03
γ - DF	1.00
ϵ (ϵ-greedy)	0.005

(b) Best Hyperparameters

Table 31: Parameters and Hyper-parameters: QLearning Exp-15

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

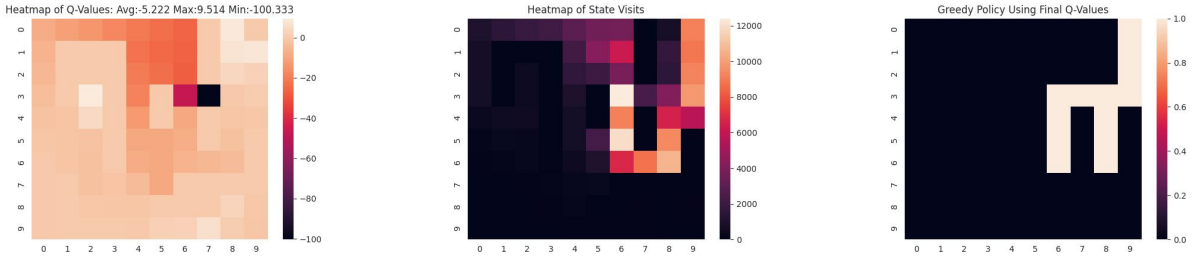


Figure 67: Heatmaps for QLearning Exp-15

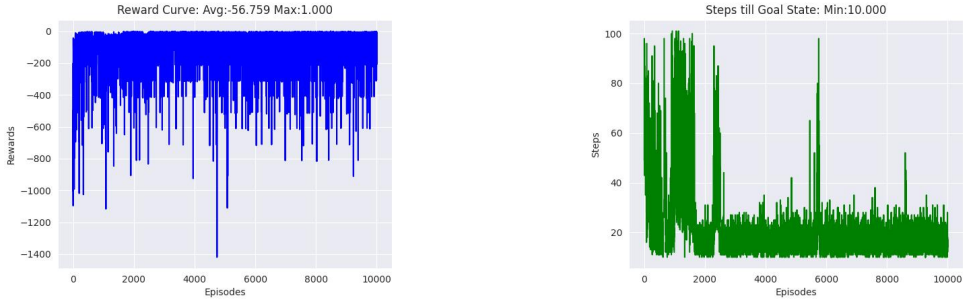


Figure 68: Plots for QLearning Exp-15

From the above plots for reward and steps, it is clear that that the RL agent learns the optimal path from the start state S1 to the nearest goal state as given in the learnt policy, considering stochaticity in the action control.

However the presence of noisy control over the agent causes it to take a lot of unnecessary steps. A wrong transition which may happen with a probability 0.3 will lead to an added penalty of 2 extra steps. This has a cumulative effect which makes it very difficuly to lower the number of steps taken beyond a certain limit.

Once again it is important to note that the start state is right next to a restart state. Thus there is a 30% chance that the first move leads to a -100 reward regardless of what action is chosen (up and down are the only possible movements here). This explains the abnormally high penalties which do not change well into training.

We can see the effect of the wind in the last leg of the agent's journey where the wind pushes it to the right and causes it to pass through the bad state incurring extra penalty. The heatmap also tells us a similar story.

8.16 Experiment 16: [S2, Windy, Noisy, Smx]

The configuration for this variant of the grid-world problem is given in the below table, along with the best combination of hyperparameters obtained after finetuning:

Parameter	Config	Code
Start State	(3,6)	S2
Wind	True	Windy
Transition Probability	0.70	Noisy
Exploration Policy	Softmax	Smx

(a) Problem Configuration

Hyperparameter	Best Value
α - LR	0.1
γ - DF	1.0
β - temperature	0.001

(b) Best Hyperparameters

Table 32: Parameters and Hyper-parameters: QLearning Exp-16

The required results of plots for reward and steps as well as the heatmaps and the learnt policy have been plotted below:

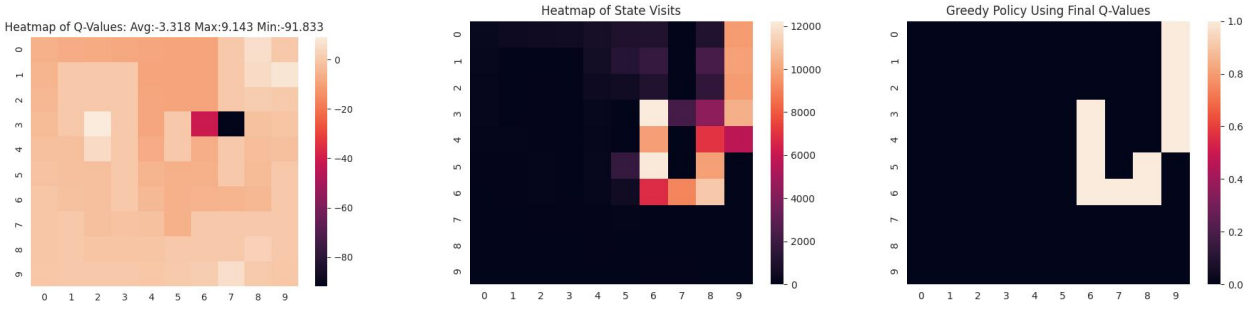
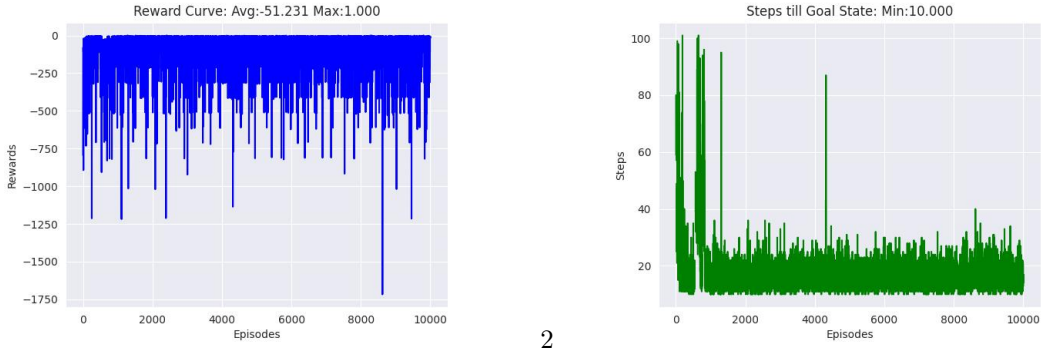


Figure 69: Heatmaps for QLearning Exp-16



2

Figure 70: Plots for QLearning Exp-16

From the above plots for reward and steps, it is quite clear that the case is similar to Experiment 15. The noisy nature of actions make it very difficult for the agent to learn.

9 Conclusion

SARSA and Q-Learning are two very popular TD algorithms. In this assignment, we performed extensive experimentation and evaluation of these control algorithms. Some of the key takeaways from this endeavour are listed below:

- In most of the deterministic ($p = 1.0$) settings, both SARSA and Q-Learning perform very well and are able to recover an optimal policy.
- Introduction of wind complicates the problem, however both algorithms are still able to generate good policies even with the stochastic event of wind.
- Both the algorithms find learning with stochastic outcomes of actions ($p = 0.7$) rather difficult as one would expect. In such cases, the policy learnt by Q-Learning is less risk-averse because it takes maximum over all actions.

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.