Programming Assignment I

REINFORCEMENT LEARNING

Aniket Khan
ME21B021

R Navin Sriram
ED21B044

February 28, 2024

Contents

1 Structure of the Project

1.1 Project Files

We divided the code into four different executable scripts, **Algorithms.py, Test.py, Gridworld.py** and **PA1.ipynb** for the sake of maintaining an easy and simple interface in the end, which would come to be very handy during the Hyperparameter Tuning. One of this being the " Gridworld Environment " given in the problem statement, performing minimal to no changes on this file. The Backbone of the Project lies in the Algorithms file within the Solver class. The **Solver class** can be easily used elsewhere and contains a set of very useful member functions. **PA1** provides a very minimalist and customizable interface to the entire project, running the scripts would automatically lead to the program saving all the plots generated for each case as requested in the Project.

1.2 Hyperparameter Tuning : Bayesian Optimisation

The script **Tune**.py contains the functionality to tune for all the worlds under all the different conditions and return the best combination of Hyperparams leading to the maximum of the objective function, which is the Reward in our case. We decided to adopt Bayesian Optimisation over other methodologies such as a Genetic Algorithm, a Naive Randomised Tuning approach and other toolboxes suchs as Optuna. Bayesian optimization uses probabilistic models to select the most promising hyperparameters for optimizing the performance. It iteratively evaluates the performance of different hyperparameter configurations, updating its model based on observed outcomes, and focuses the search in regions that are likely to yield better results, thus efficiently navigating the hyperparameter space. For comparison the **Bayesian Optimisation python toolbox** in seven observations can achieve tuning similar to that of **optuna** after 50 observations of the Target function, which is the 4D curve of Reward over γ, α, τ or ϵ, depending whether the followed policy is Softmax / Epsilon greedy

- **Gamma** : Through Empirical Optimization, we observed a regular trend. For all the worlds, a higher γ always gave better rewards with better run-times compared to lower values. For this reason we considered $\gamma = 0.95$ in all our experiments. This would allow us to convert our optimisation space to a 3D space, reducing runtime.
- **Tau** : During Softmax, the hyperparameter τ or the temperature controls the degree of exploration and exploitation incurred by the policy. A high τ often means increased bias towards exploration and a lower indicates bias towards Exploitation.
- **Espilon** : During Epsilon Greedy, the hyperparameter ϵ or the temperature controls the degree of

exploration and exploitation incurred by the policy. A high ϵ often means increased bias towards exploration and a lower indicates bias towards Exploitation.

- **Alpha** : With a high learning rate, the agent gives more weight to new information compared to the existing estimates. From the experiments we noticed that this often led to quick convergence but with a lot of oscillations and overshoots whereas in comparison, a lower α converges slowly and smoothly.

1.3 Plots

- **Reward curves** and the **number of steps** to reach the goal in each episode.
- **Heatmap of the grid with state visit counts**, i.e., the number of times each state was visited throughout the training phase averaged over total number of episodes.
- **Heatmap of the grid with Q values** after training is complete, and optimal actions for the best policy

1.4 Choice of Policy

Softmax was chosen as the governing policy over Epsilon greedy. Epsilon Greedy with the optimised parameters achieved costs almost in the vicinity of softmax if not higher after the optimization for tuning the Hyperparameters. The computation time remained similar for both the cases, due to this Softmax was chosen because Softmax tends to better utilize the learned information, as it assigns probabilities to actions based on their relative values. This can lead to more efficient learning and exploitation of the environment.

1.5 Tuned Hyper Params

Sarsa

- Learning Rates Softmax

$$\begin{bmatrix}0.417022004 & 0.4170220047 & 0.302044776 & 0.302044776 & 0.41702200 & 0.302044776\end{bmatrix}$$

- Learning Rates Eps Greedy

$$\begin{bmatrix}0.4385967 & 0.417022004 & 0.0318100 & 0.01209676 & 0.012096760 & 0.4170220\end{bmatrix}$$

- Temperatures

$$\begin{bmatrix}0.74829204 & 0.748292044 & 0.93674488 & 0.74829204 & 0.74829204 & 0.752508895\end{bmatrix}$$

- Espilon

$$\begin{bmatrix}0.66593058 & 0.369319122 & 0.374574766 & 0.72830833 & 0.3745747 & 0.66593058\end{bmatrix}$$

Q Learning

- Learning Rates softmax

$$\begin{bmatrix} 0.44284451 & 0.41500703 & 0.4150070363 & 0.41500703 & 0.086072098 & 0.462291208 \end{bmatrix}$$

- Learning Rates Eps greedy

$$\begin{bmatrix} 0.149306713173 & 0.015691638 & 0.01556920320 & 0.42657081177004946 & 0.01556920 & 0.149306713 \end{bmatrix}$$

- Temperatures

$$\begin{bmatrix} 0.69425138 & 0.75250889 & 0.752508895 & 0.7525088958 & 0.782822203806 & 0.6536969925 \end{bmatrix}$$

- Espilon

$$\begin{bmatrix} 0.7031411357 & 0.7482920440 & 0.37154284 & 0.373011104 & 0.3730111 & 0.7482920440 \end{bmatrix}$$

2

Experiment 1

2.1 World Parameters:

- start = [0,4]
- wind = False
- deterministic step (p=1)

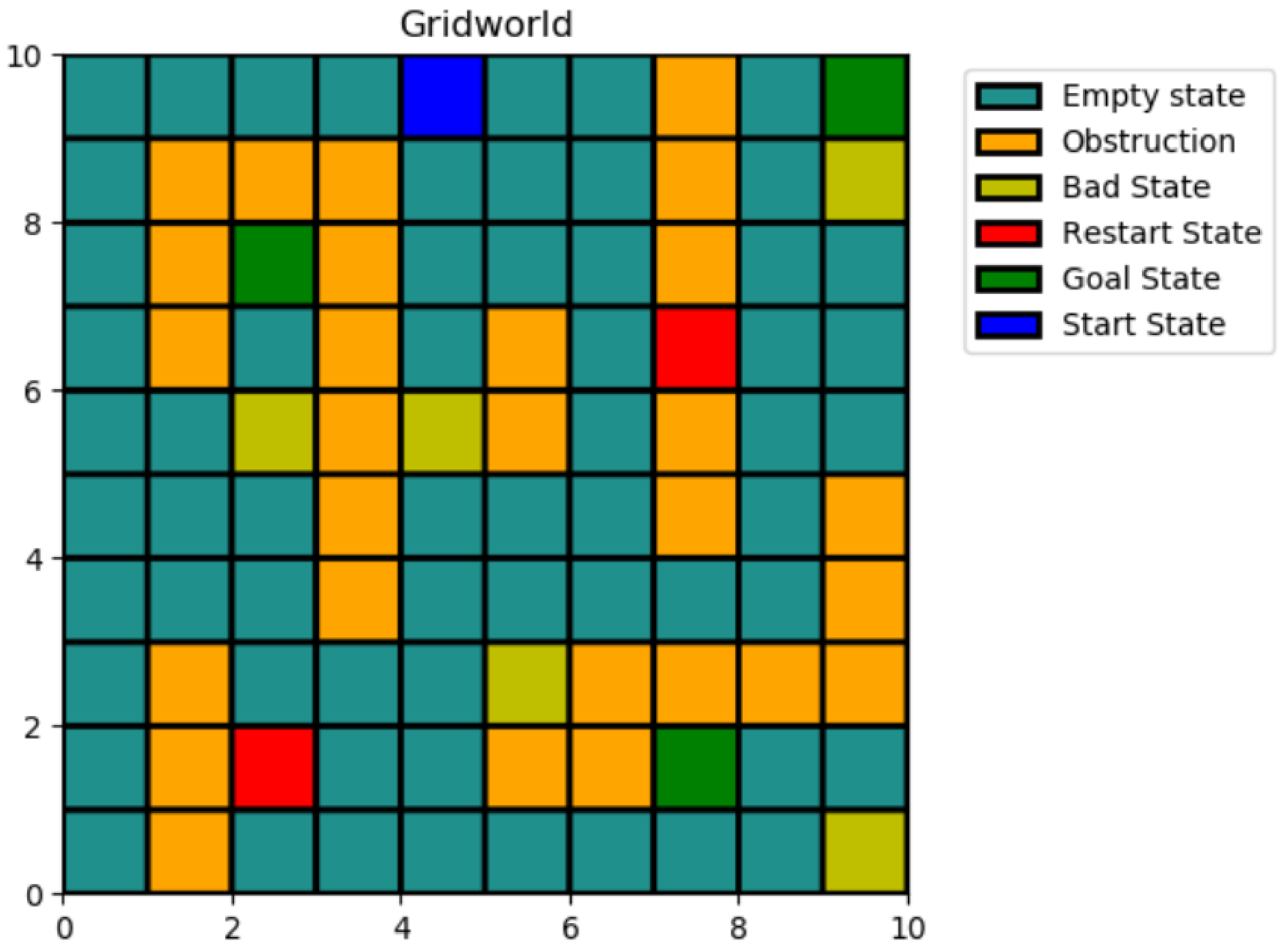
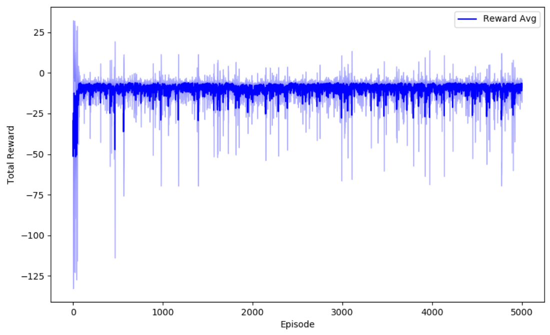


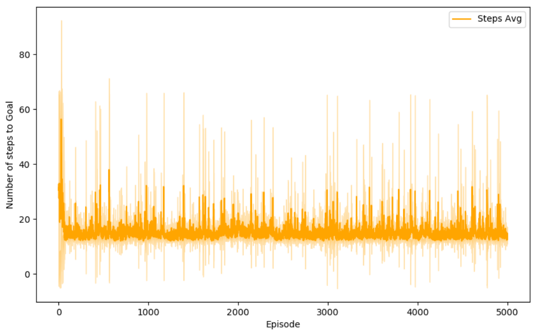
Figure 1: World 1

2.2 Sarsa

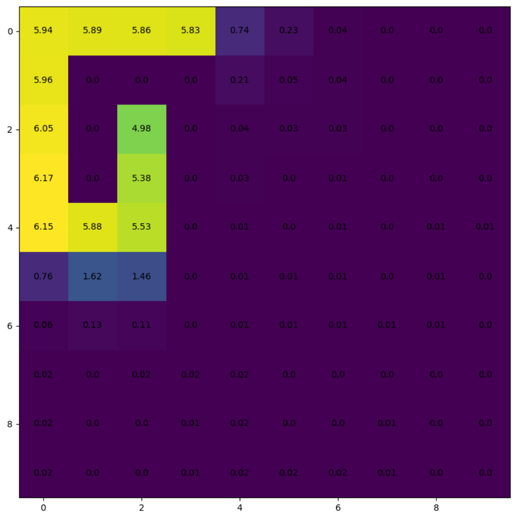
- **Policy:** Softmax
- $\alpha = 0.41702$
- $\tau = 0.7482$
- $\gamma = 0.95$

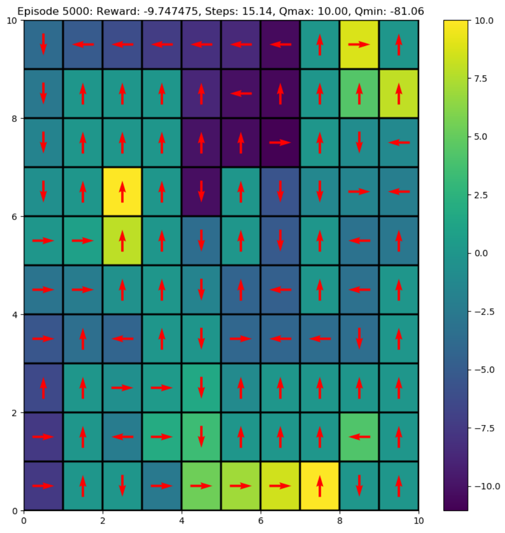


(a) Average reward VS Episodes



(b) Average steps vs Episodes


(c) Heatmap of state with Visit Count

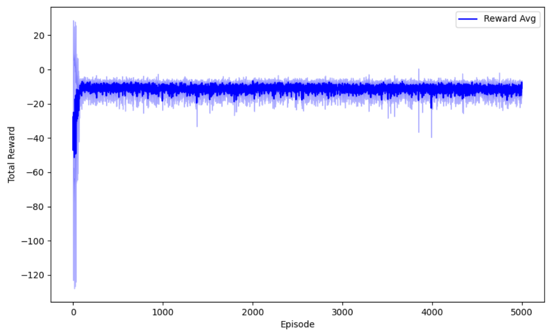


(d) Heatmap of state with Q values

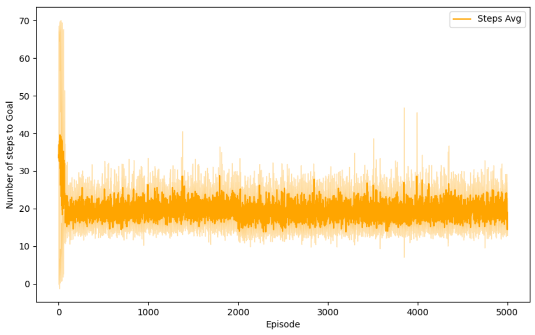
Figure 2: Plots for SARSA algorithm with softmax policy

2.3 QLEARNING

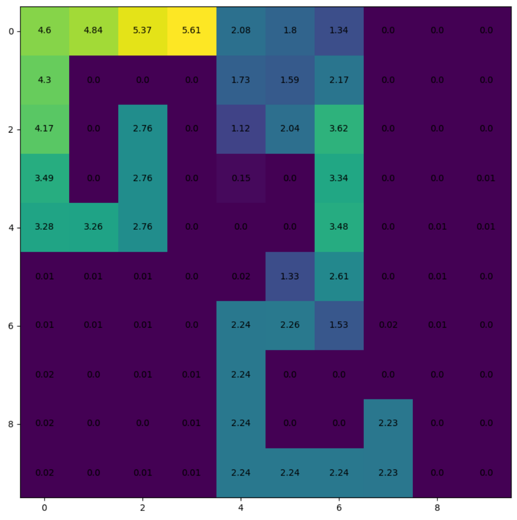
- **Policy:** Softmax
- $\alpha = 0.44284$
- $\tau = 0.6942$
- $\gamma = 0.95$

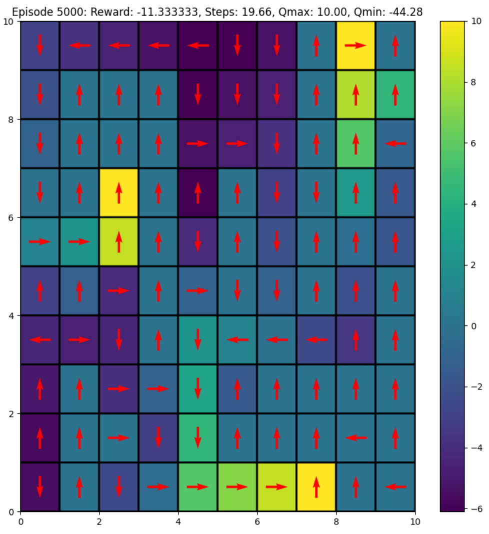


(a) Average reward VS Episodes



(b) Average steps vs Episodes


(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 3: Plots for QLearning algorithm with softmax policy

3 EXPERIMENT 2

3.1 WORLD PARAMETERS:

- start = [0,4]
- wind = False
- stochastic step (p=0.7)

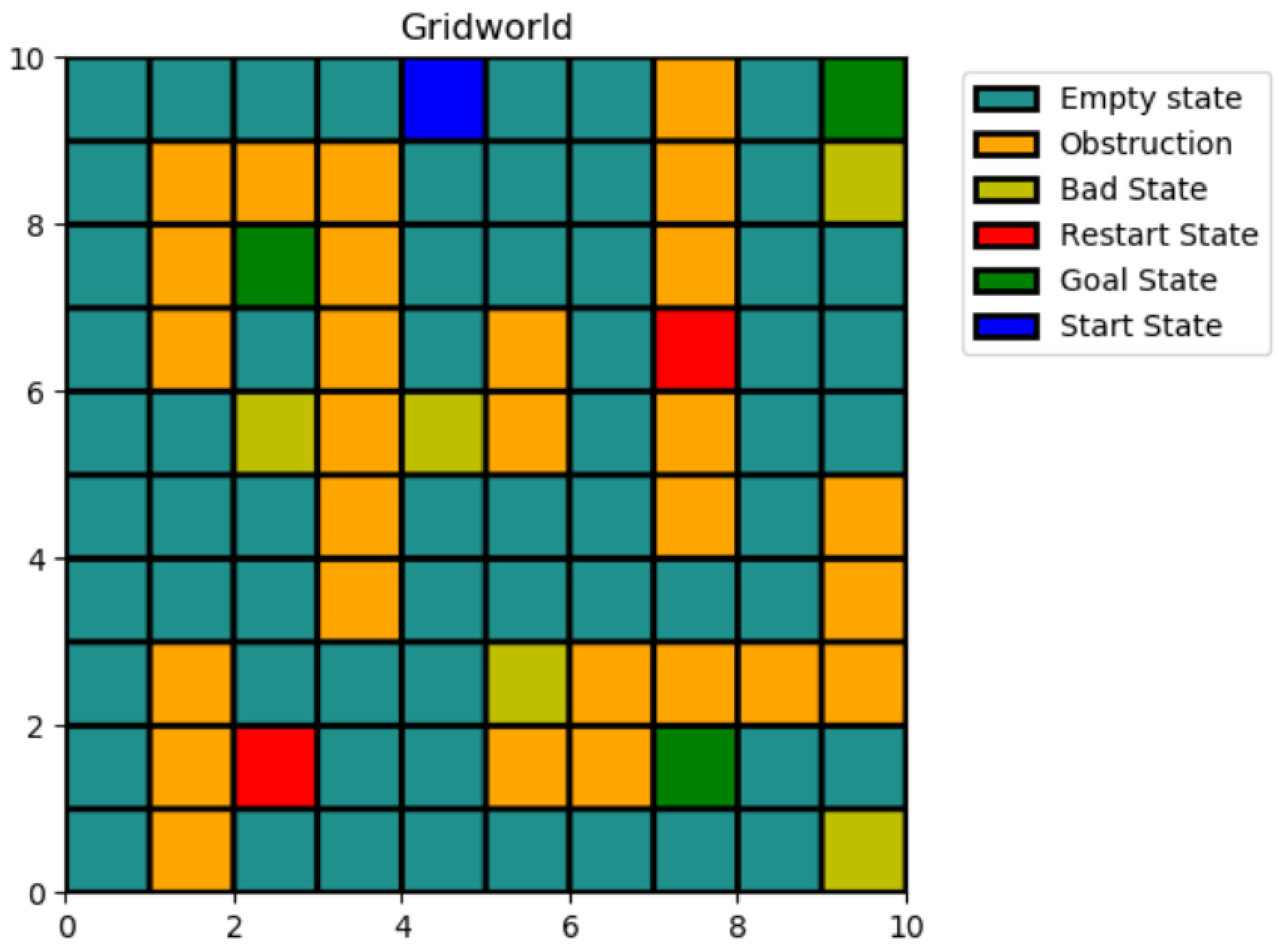
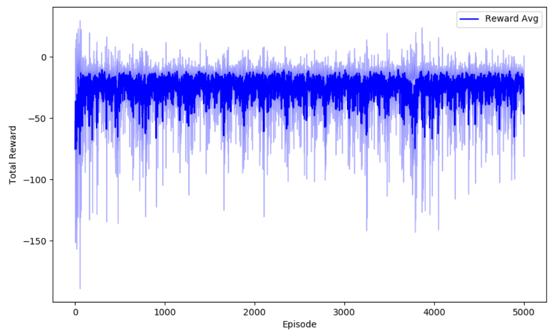


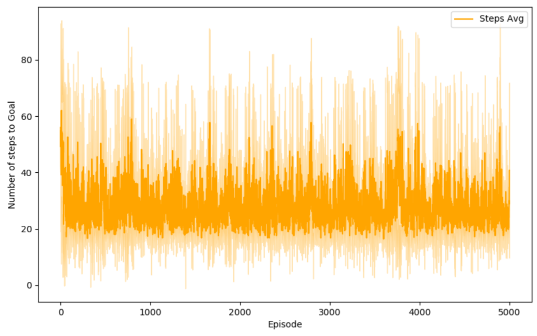
Figure 4: World 2

3.2 Sarsa

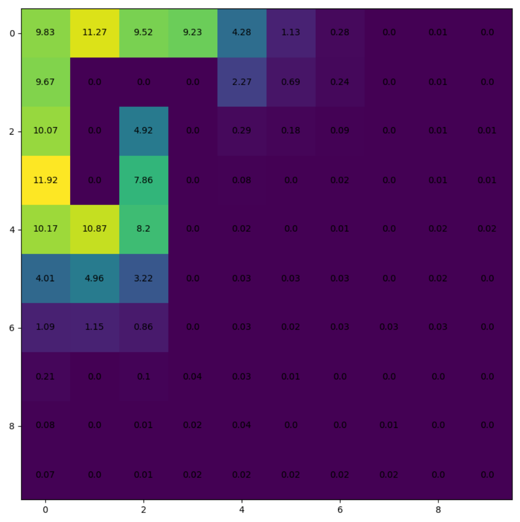
- **Policy:** Softmax
- $\alpha = 0.41702$
- $\tau = 0.7482$
- $\gamma = 0.95$



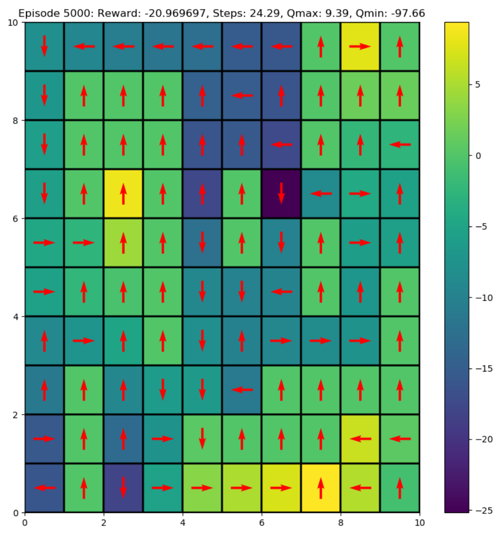
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count

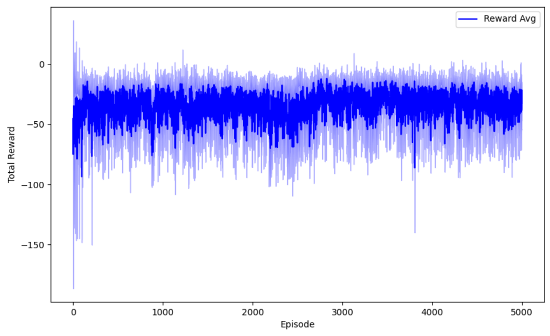


(d) Heatmap of state with Q values

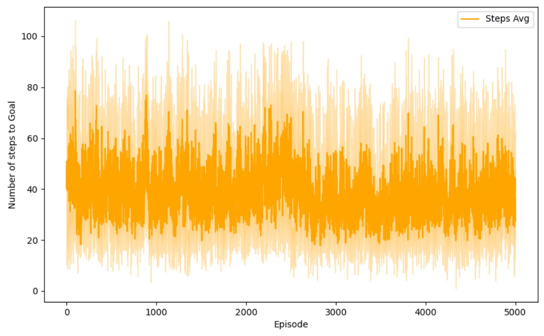
Figure 5: Plots for SARSA algorithm with softmax policy

3.3 QLEARNING

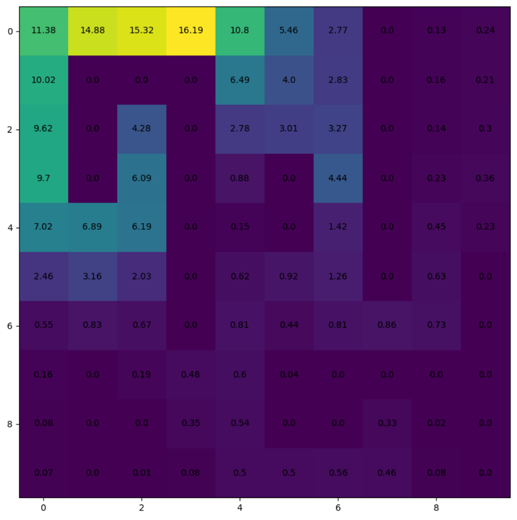
- **Policy:** Softmax
- $\alpha = 0.41500$
- $\tau = 0.7525$
- $\gamma = 0.95$



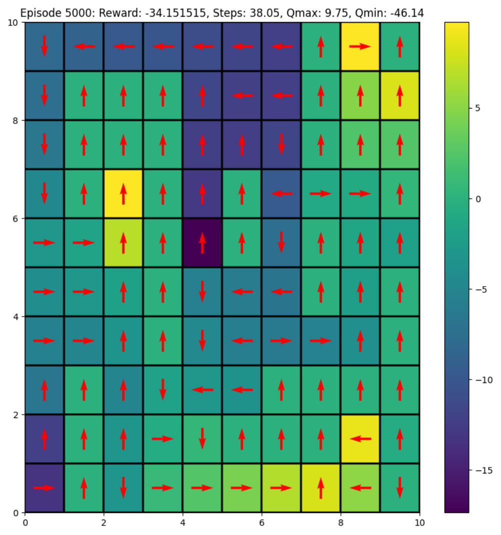
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 6: Plots for QLearning algorithm with softmax policy

4 Experiment 3

4.1 World Parameters:

- start = [0,4]
- wind = True
- Deterministic step (p=1)

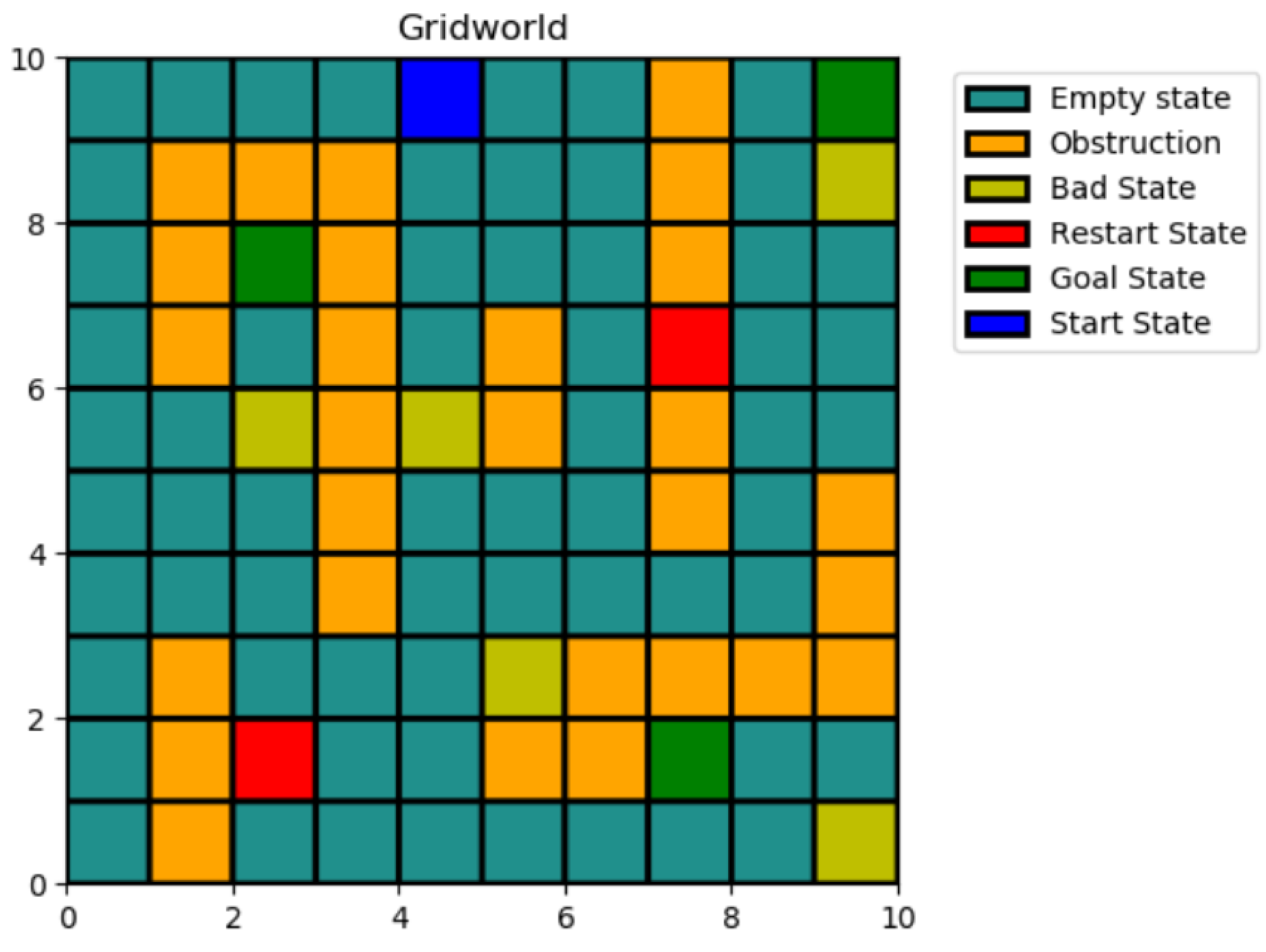
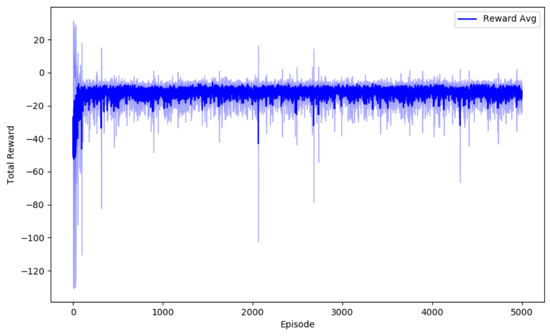


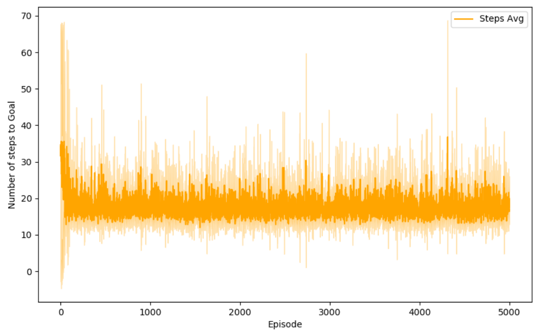
Figure 7: World 3

4.2 SARSA

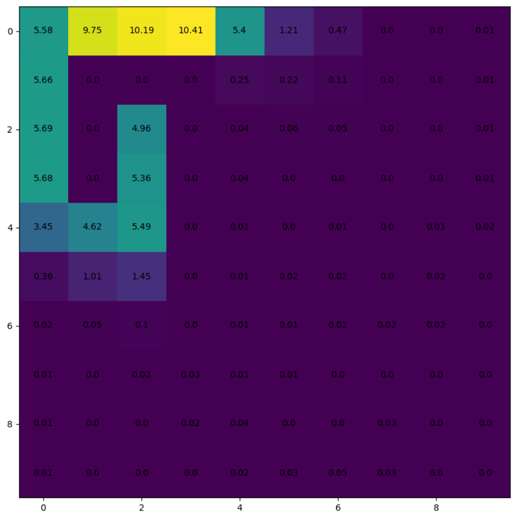
- **Policy:** Softmax
- α = 0.30204
- τ = 0.9367
- γ = 0.95



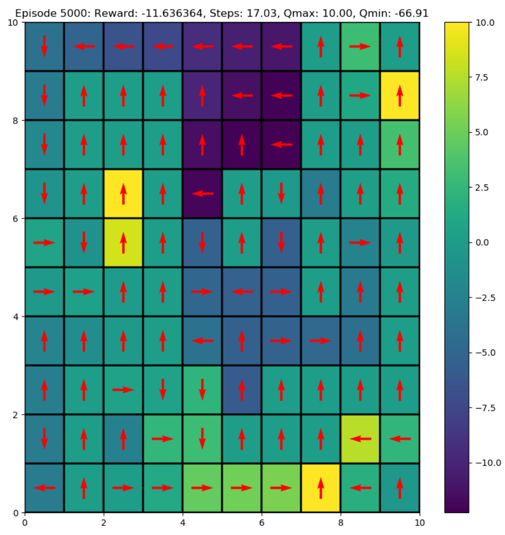
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count

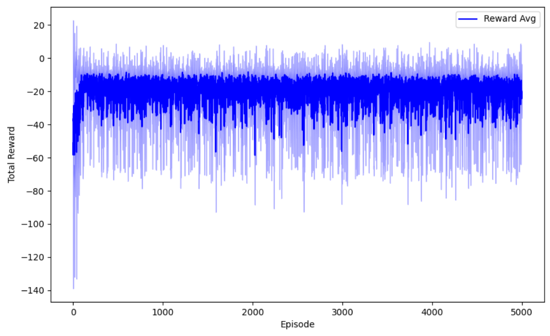


(d) Heatmap of state with Q values

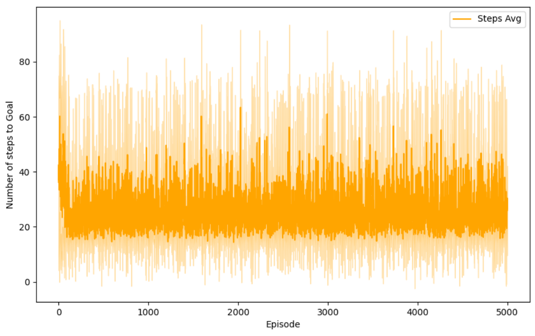
Figure 8: Plots for SARSA algorithm with softmax policy

4.3 QLEARNING

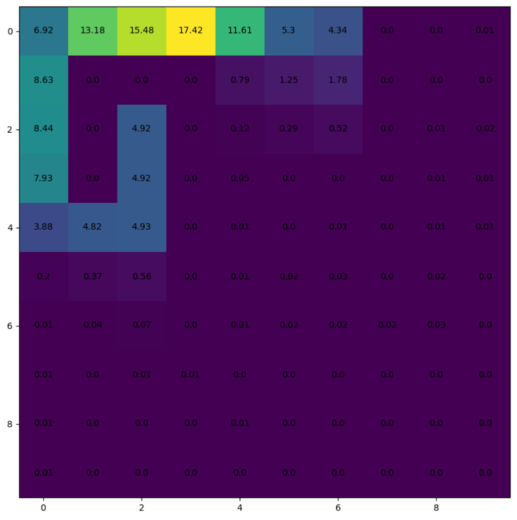
- **Policy:** Softmax
- $\alpha = 0.41500$
- $\tau = 0.7525$
- $\gamma = 0.95$



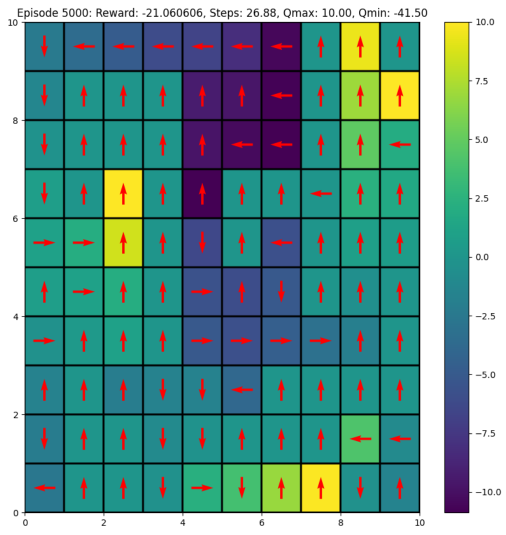
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 9: Plots for QLearning algorithm with softmax policy

5

Experiment 4

5.1 World Parameters:

- start = [3,6]
- wind = False
- Deterministic step (p=1)

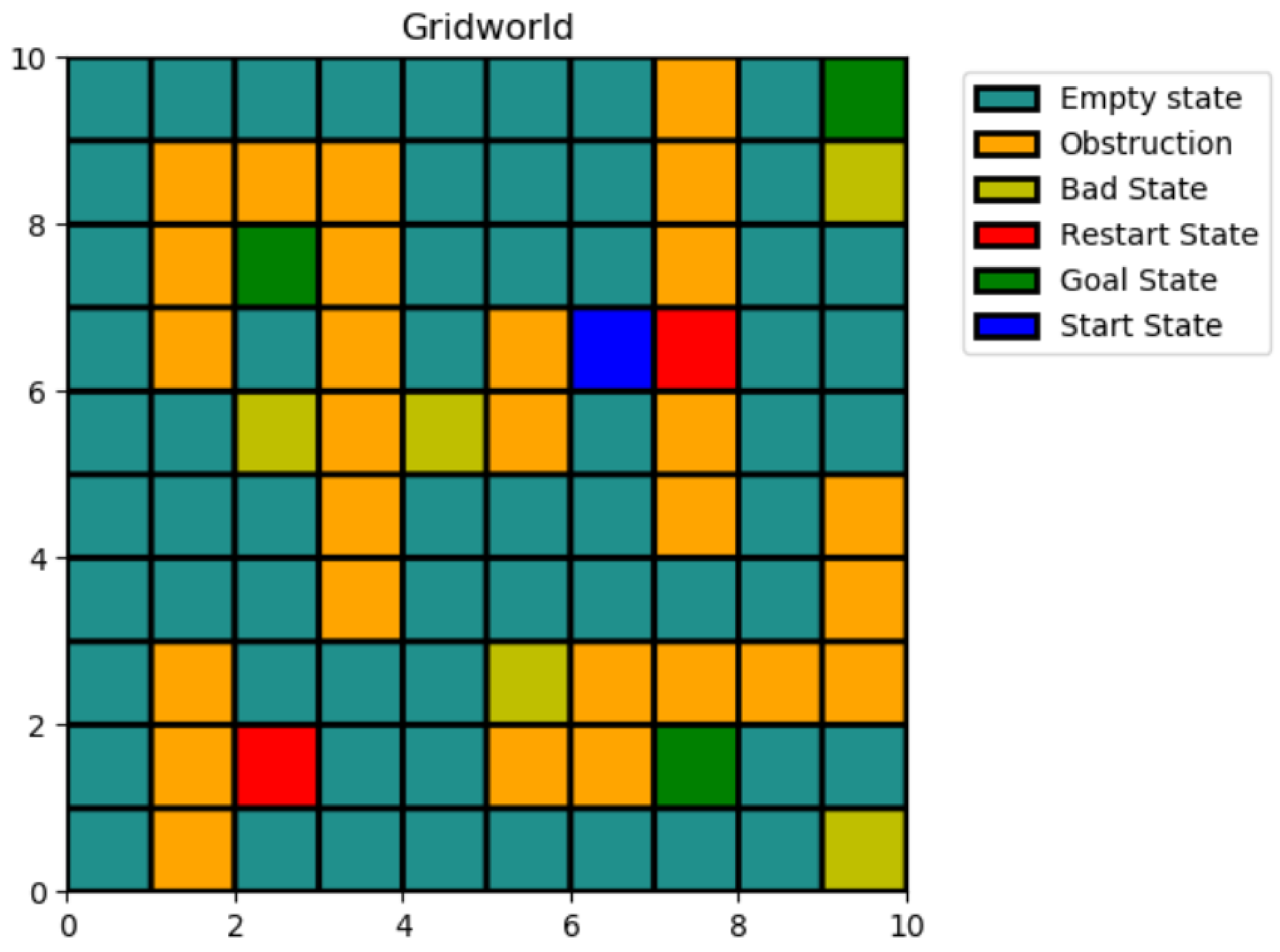
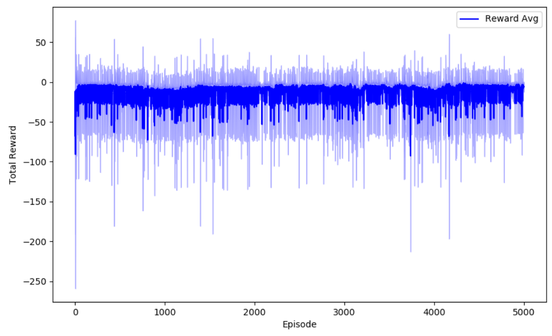


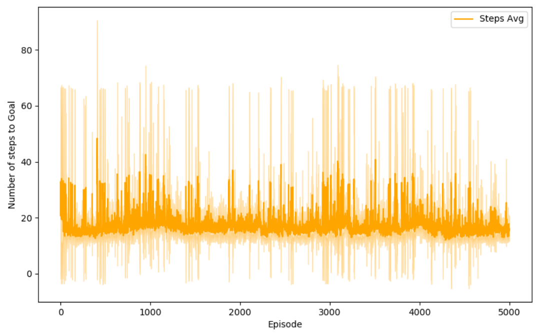
Figure 10: World 4

5.2 Sarsa

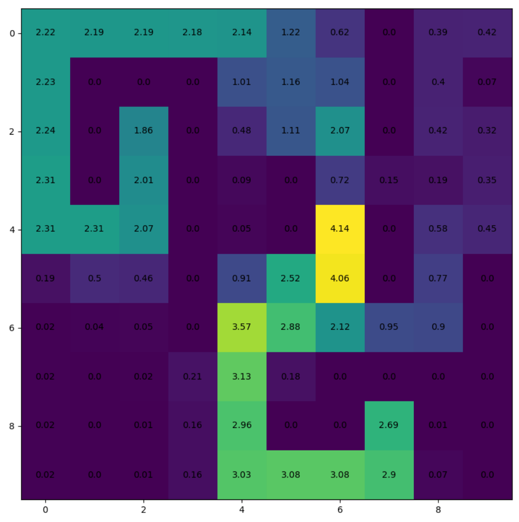
- **Policy:** Softmax
- $\alpha = 0.30204$
- $\tau = 0.7482$
- $\gamma = 0.95$



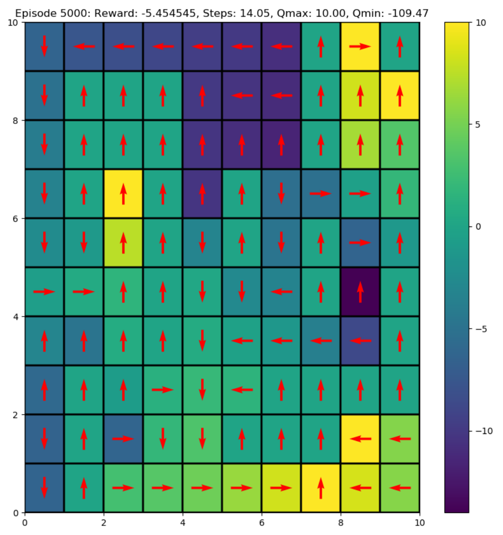
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count

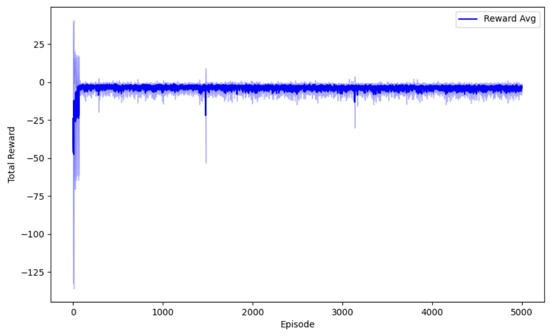


(d) Heatmap of state with Q values

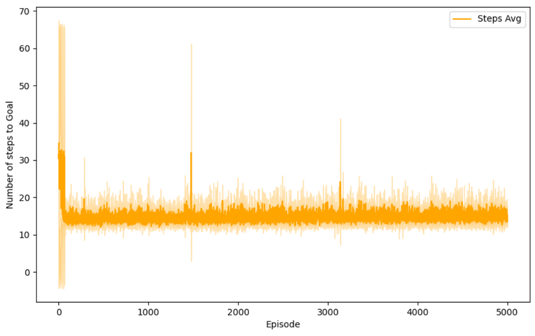
Figure 11: Plots for SARSA algorithm with softmax policy

5.3 QLEARNING

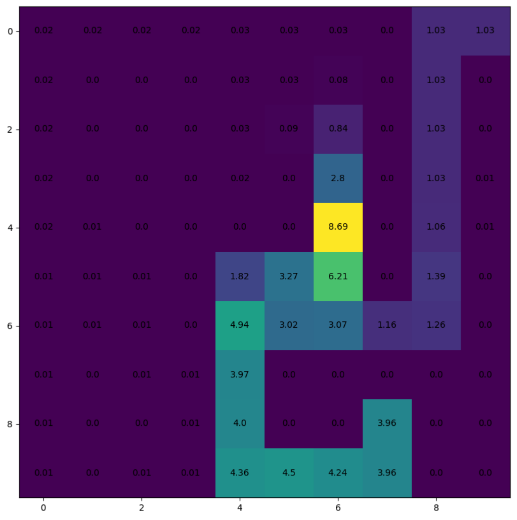
- **Policy:** Softmax
- $\alpha = 0.41500$
- $\tau = 0.7525$
- $\gamma = 0.95$



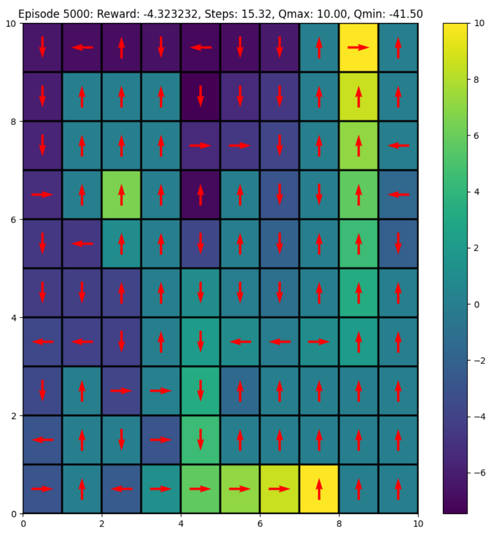
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 12: Plots for QLearning algorithm with softmax policy

6 Experiment 5

6.1 World Parameters:

- start = [3,6]
- wind = False
- Stochastic step (p=0.7)

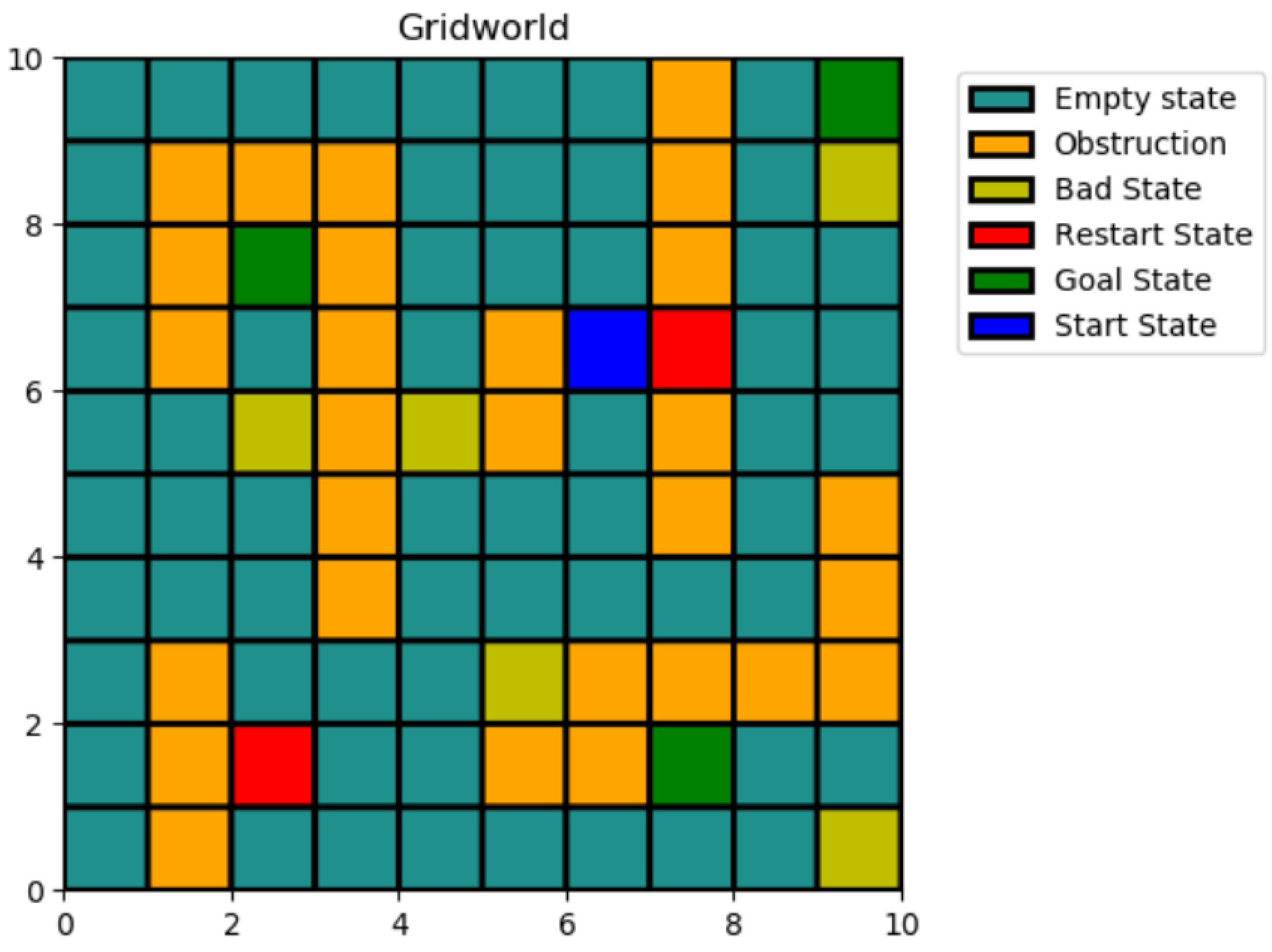
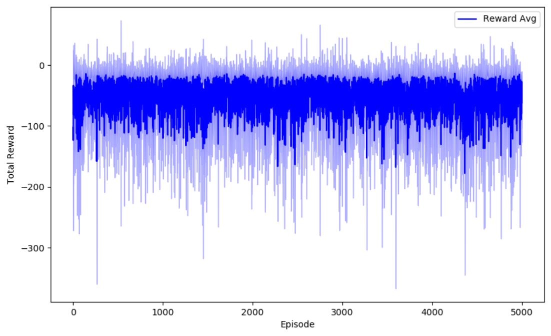
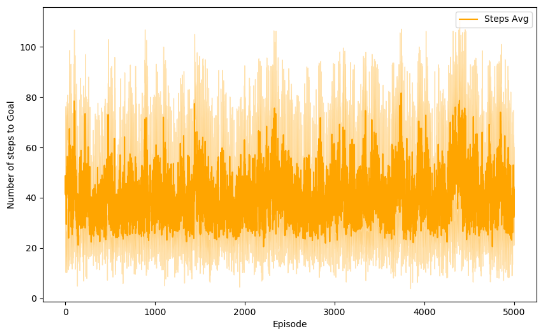


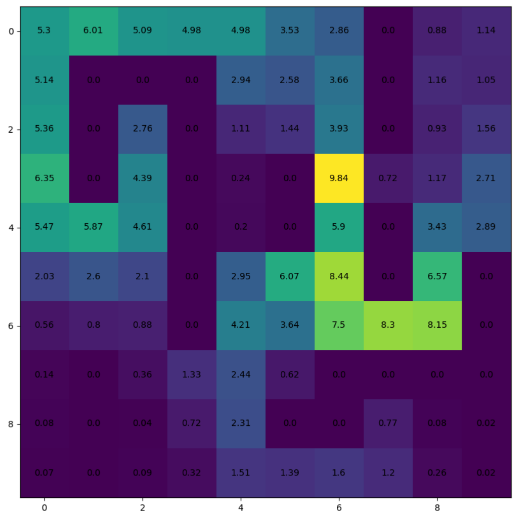
Figure 13: World 5

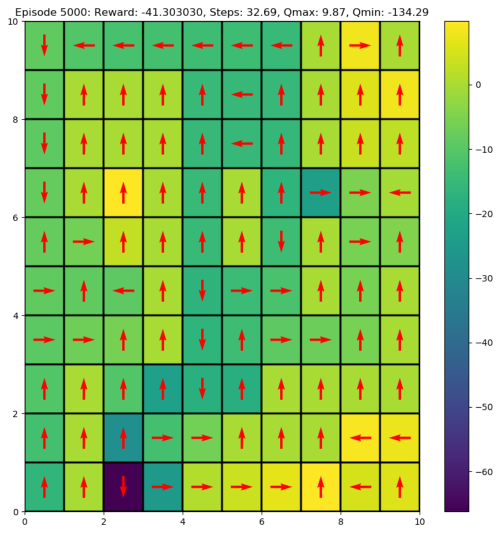
6.2 Sarsa

- **Policy:** Softmax
- $\alpha = 0.41702$
- $\tau = 0.7482$
- $\gamma = 0.95$


(a) Average reward VS Episodes


(b) Average steps vs Episodes


(c) Heatmap of state with Visit Count

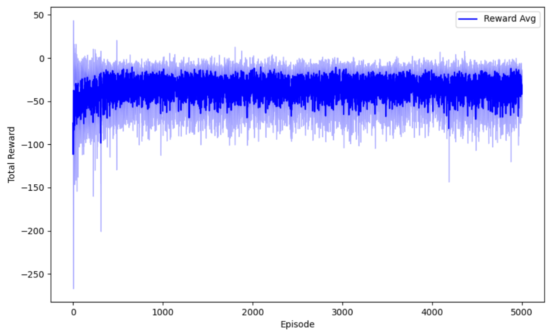


(d) Heatmap of state with Q values

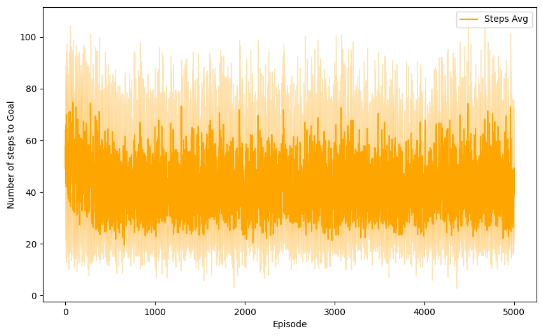
Figure 14: Plots for SARSA algorithm with softmax policy

6.3 QLEARNING

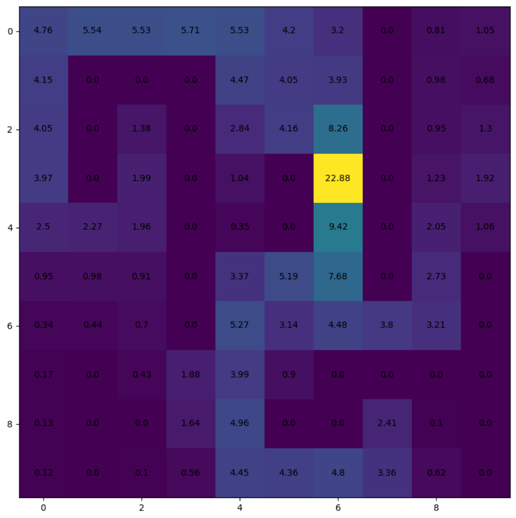
- **Policy:** Softmax
- $\alpha = 0.08607$
- $\tau = 0.7828$
- $\gamma = 0.95$



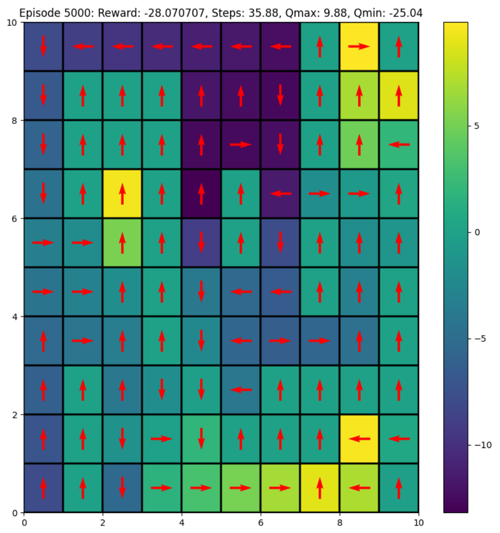
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 15: Plots for QLearning algorithm with softmax policy

7 Experiment 6

7.1 World Parameters:

- start = [3,6]
- wind = True
- Detereministic step (p=1)

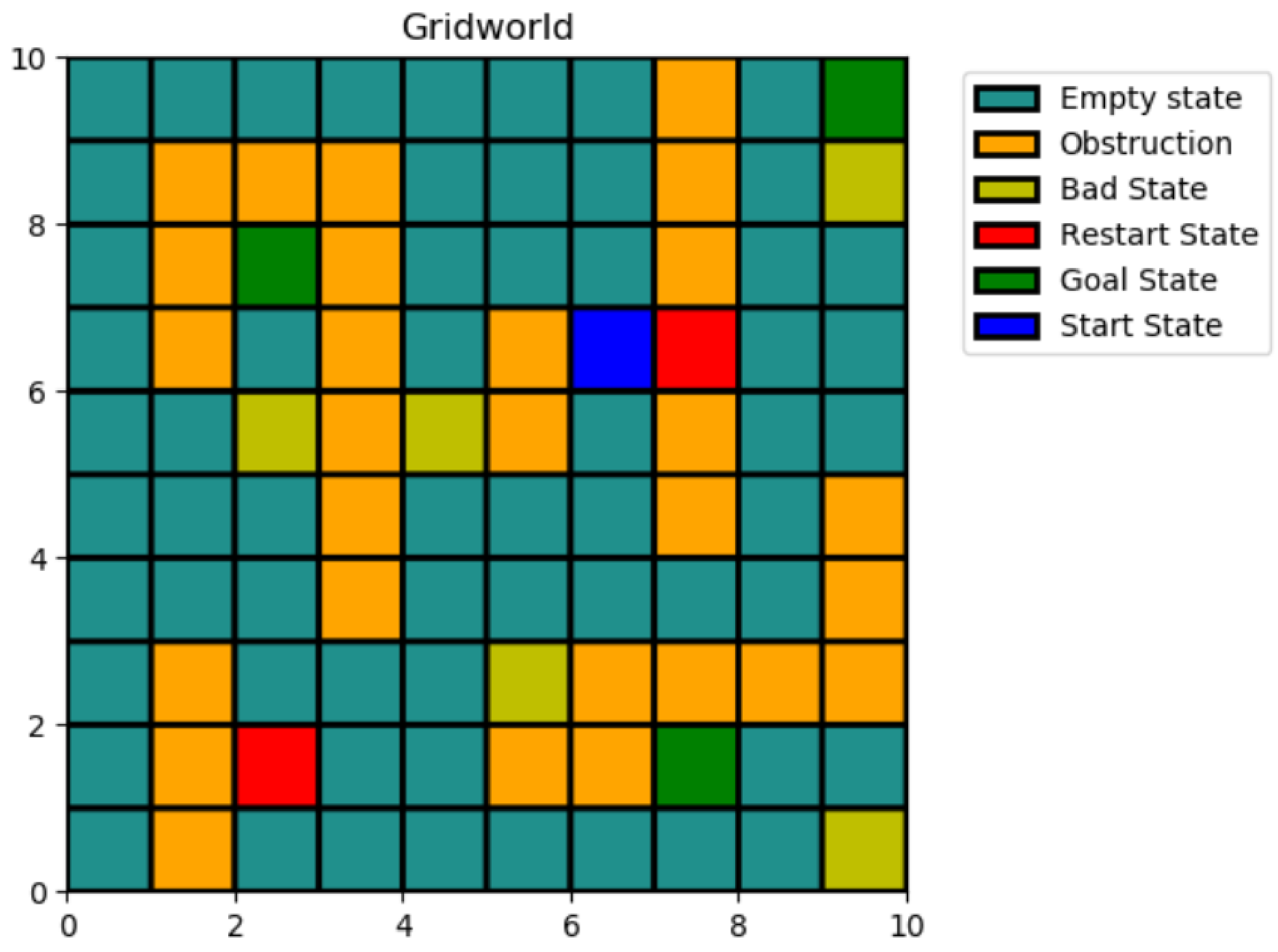
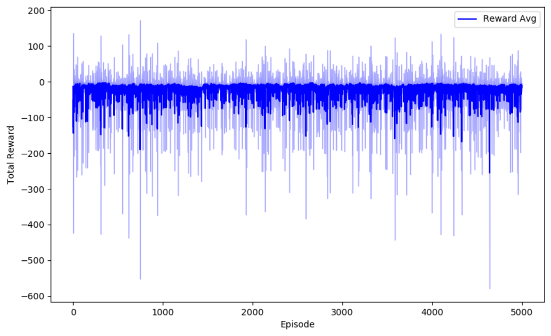


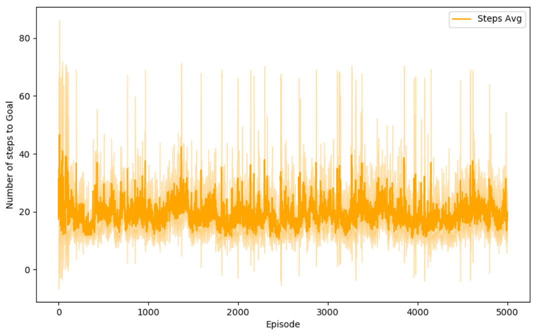
Figure 16: World 6

7.2 Sarsa

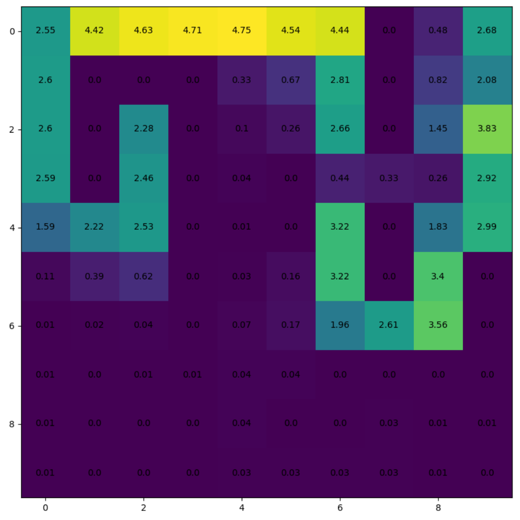
- **Policy:** Softmax
- $\alpha = 0.30204$
- $\tau = 0.7525$
- $\gamma = 0.95$



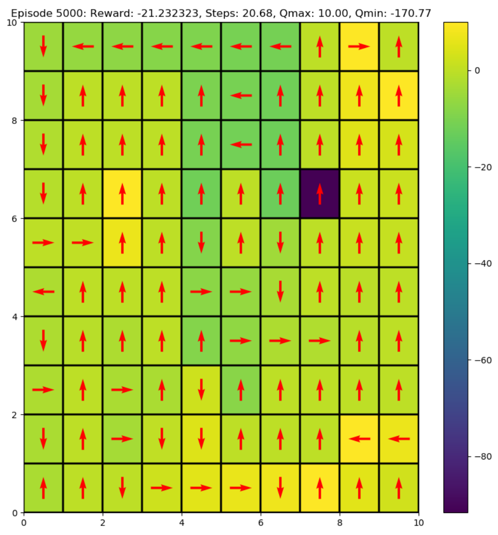
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count

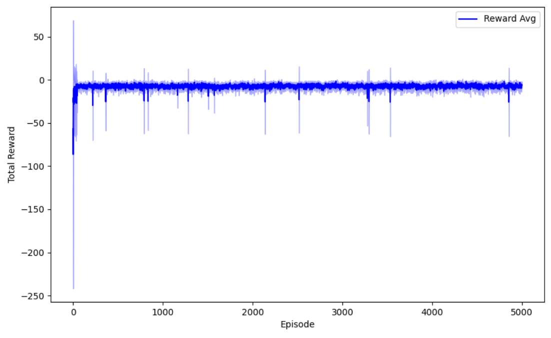


(d) Heatmap of state with Q values

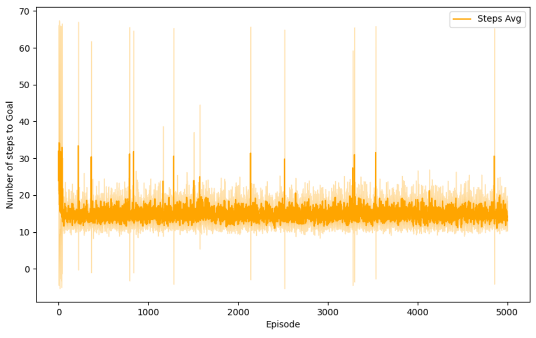
Figure 17: Plots for SARSA algorithm with softmax policy

7.3 QLearning

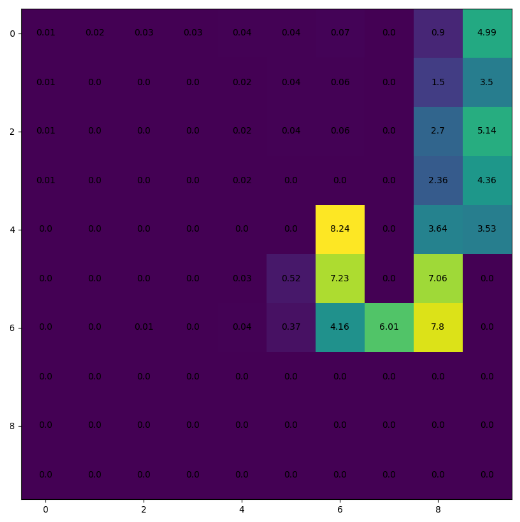
- **Policy:** Softmax
- $\alpha = 0.46229$
- $\tau = 0.6536$
- $\gamma = 0.95$



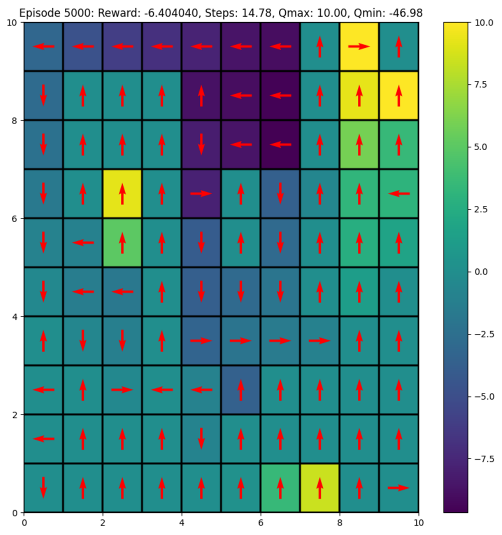
(a) Average reward VS Episodes



(b) Average steps vs Episodes



(c) Heatmap of state with Visit Count



(d) Heatmap of state with Q values

Figure 18: Plots for QLearning algorithm with softmax policy

8

Inference and Conclusion

World 1

- **A**chieved average rewards in the vicinity of -9.7 and steps in the range of 16-17 which is near optimal.
- Overshoots observed in the reward and steps waveform due to the significantly large alpha.
- Visit counts show the optimality of the generated policy with the erroneously large number of visits along the shortest path
- Significantly more overshoots can be observed in the case of Q learning. Q learning is more focused on exploitation by going off-policy by choosing the optimal policy instead of the evaluation policy.

World 2

- **A**chieved average rewards in the vicinity of -20 and steps in the range of 25 which is significantly worse than the first world. This can be largely attributed to the stochasticity introduced in going to the next cell.
- Visit counts still show the optimality of the generated policy with the erroneously large number of visits along the shortest path
- Significantly more and more overshoots can be observed in the case of Q learning. Q learning is more focused on exploitation by going off-policy by choosing the optimal policy instead of the evaluation policy.

World 3

- Achieved average rewards in the vicinity of -20 and steps in the range of 25 which is significantly worse than the first world. This can be largely attributed to the stochasticity introduced in going to the next cell due to the wind
- The increased deviations in Q learning remain a common feature for all the experiments

World 4

- Achieved best rewards in the range of -4 and steps in the range of 19 which is significantly worse than the first world. This can be largely attributed to the stochasticity introduced in going to the next cell due to the wind
- Due to the change in the start position, in the new configuration there are multiple goal states in the vicinity.
- The policy tries to reach different goal states during different iterations, leading to a large deviations in the step count and a more colourful visit count heat map

World 5

- This is similar to world 2 except for the start state. Rewards are worse compared to World 2 and World 4 owing to the stochasticity and presence of multiple goal locations, making the algorithm explore all the paths greedily during some iterations
- The policy tries to reach different goal states during different iterations, leading to a large deviations in the step count and a more colourful visit count heat map

World 6

- This is similar to world 3 except for the start state. Rewards are worse compared to World 3 and World 4 owing to the stochasticity due to the wind and presence of multiple goal locations, making the algorithm explore all the paths greedily during some iterations
- The policy tries to reach different goal states during different iterations, leading to a large deviations in the step count and a more colourful visit count heat map
- On observing we notice that the behaviour of both the algorithms remain more or less the same in the stochastic step and wind case