# CS 747: Assignment 3

Mithilesh Vaidya
17D070011

## 1 Task 1, 2, 3, 4

### 1.1 Design

- The 7x10 grid and the wind in each column from Example 6.5 in Sutton and Barto [1] is chosen for this set of experiments.
- As mentioned in [1], learning rate $\alpha = 0.5$ while $\epsilon = 0.1$. Results are averaged out over 15 seeds and the agent learns over 200 episodes.
- The reward for reaching the final state is set to 1 while reward is -1 for reaching any other state.
- In case of wind at one of the edges: First, the final state is calculated without the constraints of the grid. Then, the state is clipped so that the agent lies within the grid.
- In Task 4, the convention for stochastic wind is similar to the aforementioned convention: The combination of the agent's action and the stochastic wind is first calculated. If it lies beyond the boundaries of the grid, the agent is constrained to the closest location in the grid from this new state which is outside the grid.

### 1.2 Observations

Individual and combined plots are given below. The trend is as expected:

- The performance of the 4 move agent acts as a baseline to compare the other two plots.

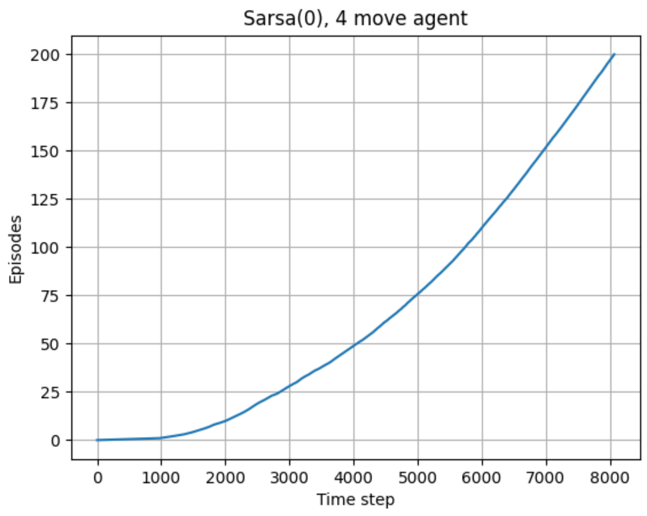

Figure 1: Sarsa(0) in case of 4-move agent.

- The agent performs the best in case of 8 moves since it gives more flexibility to choose the next state. Diagonal moves helps the agent reach various locations in the grid in fewer steps than when the agent is only limited to 4 moves. The simplest example is that of moving diagonally from a location. In case of only 4 moves, the agent will require 2 steps but in case of 8 moves, the agent can reach the desired location in just one step.

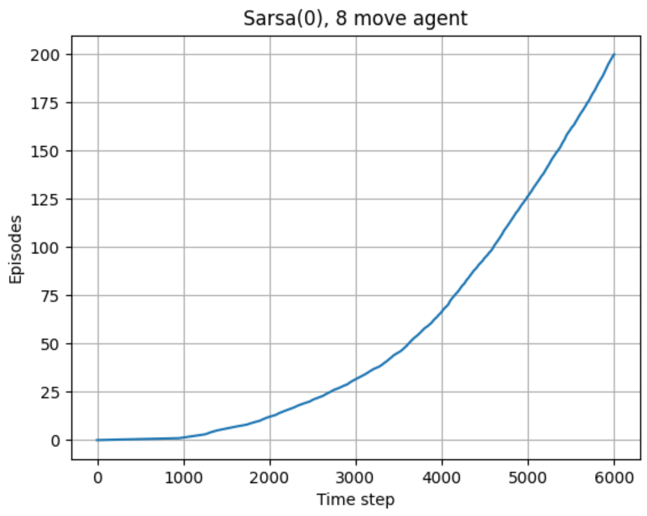

Figure 2: Sarsa(0) in case of 8-move agent.

- On the other hand, 8 moves + stochastic wind performs worse than even the 4 move agent since the presence of stochastic wind makes it harder for the agent to move to a desired location. The agent's prediction of the action-value function will be noisy. Hence, the agent needs more steps to reach the end state.

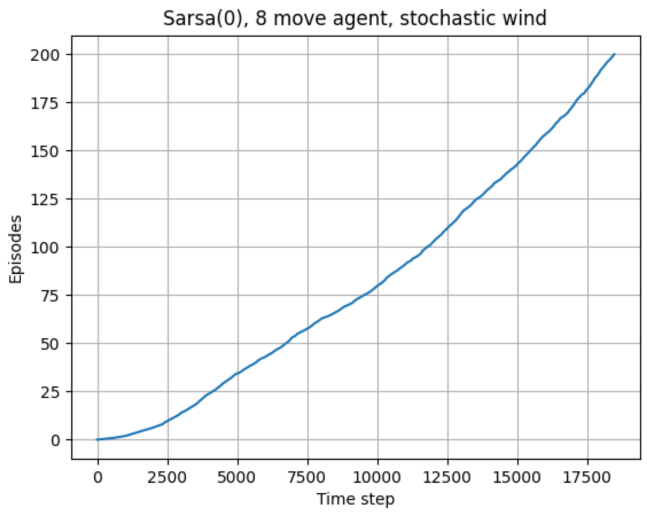

Figure 3: Sarsa(0) in case of 8 moves + stochastic wind.

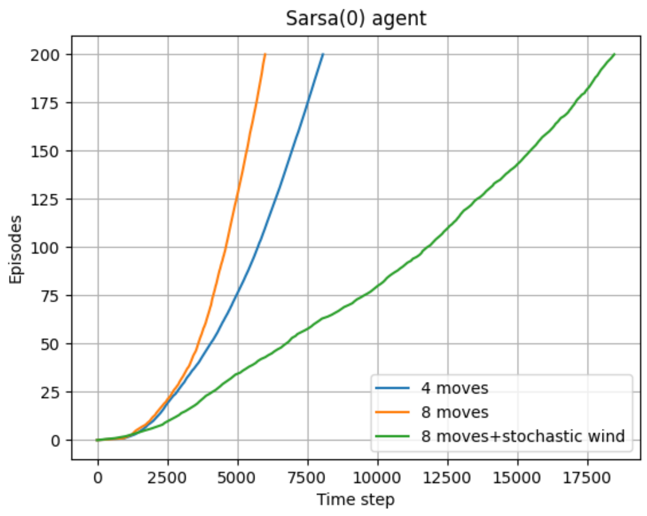

Figure 4: Comparison of the above 3 cases.

- The increasing slope of the *Episode vs Time Step* in case of 4 move and 8 move without stochastic wind implies that less time steps are required to complete a episode as the agent learns. This indicates that the agent successfully learns to navigate the grid and reach the end state.
- In case of the 8 move + stochastic wind, the agent learns in the beginning but the smaller slope at the end implies that it takes many more time steps to reach the terminal state as compared to the other 2 environments.

## 2 Task 5

### 2.1 Design

- In this task, we compare the performance of Sarsa, Expected Sarsa and Q-Learning in case of the 4-move world without stochastic wind.
- The learning rate and epsilon are unchanged: $\alpha = 0.5$, $\epsilon = 0.1$. Number of episodes and number of independent trials is also kept the same.

### 2.2 Observations

Plot is given in figure 5.

- The order of performance is: Q-Learning > Expected Sarsa > Sarsa.
- Since the Q-Learning update is off-policy, it can quickly learn the optimal policy as compared to the other two.

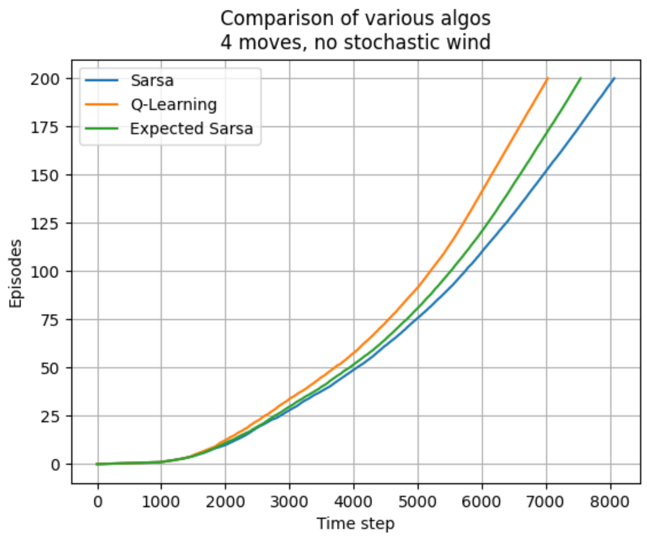

Figure 5: Comparing the performance of Sarsa, Expected Sarsa and Q-Learning

- Expected Sarsa moves in a direction which is the expectation of the direction in which Sarsa moves (hence the name). Although it is computationally more expensive than Sarsa, it's update is much more stable since expectation eliminates the noise.

## 3 Some more experiments

I did some additional parameter tuning to see it's effect on the performance of the agent. The simple 4-move grid is used without stochastic wind. Sarsa(0) is the algorithm. Results are reported by averaging out over 15 independent runs. Each run comprises of 200 episodes.

### 3.1 Varying alpha

In this set of experiments, I varied the learning rate $\alpha$ from 0.1 to 0.9 in steps of 0.1. Given below is the plot.
We observe that the agent performs best when $\alpha \approx 0.7$. This indicates that there is a sweet spot - not too high and not too low.

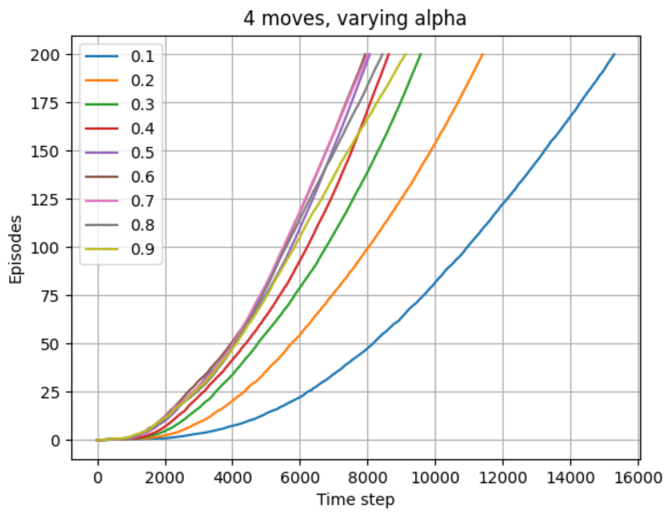

Figure 6: Varying the learning rate

## 3.2 Varying Epsilon

In this set of experiments, $\epsilon$ range covers various orders of magnitude. The corresponding performance is given below.
It is observed that the algorithm performs best when $\epsilon$ is 0. Note that Sarsa(0) for $\epsilon = 0$ moves closer to Q-Learning (sampling of arms is still different). We have already observed the superior performance of Q-Learning in task 5. As $\epsilon$ increases, the performance goes down.

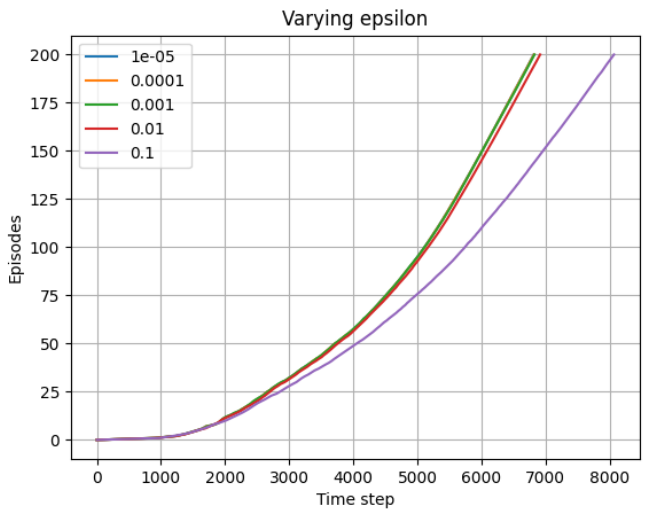

Figure 7: Varying the learning rate

## 3.3 Varying Terminal Reward

In this set of experiments, the terminal reward is varied from 0 to $10^4$. It's effect is quite drastic.
Best performance is seen when terminal reward is 1. As the reward increases, performance goes down. A possible explanation could be that the negative reward of -1, which gets accumulated as the agent moves to non-terminal states, gets dominated by the huge positive reward. As a result, the difference between Q-values of optimal and non-optimal paths is very small and hence the agent may end up moving randomly in the grid.

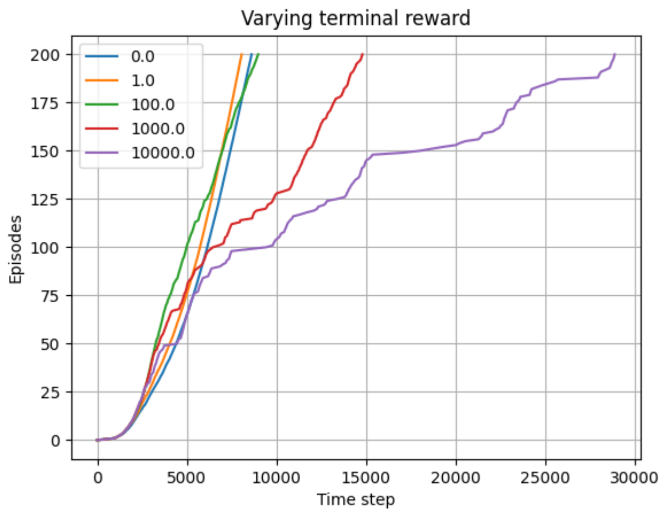

Figure 8: Varying the learning rate

## 3.4 Random Walk vs Stochastic Wind

How poor is the performance of the agent in task 4? I compared it with a random walk i.e. at each state, the agent chooses any of the 8 kings moves randomly. Results are given below:

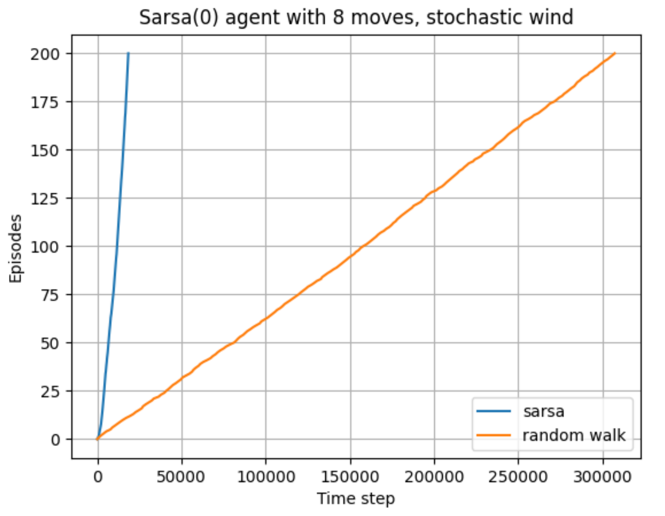

Figure 9: Sarsa(0) vs Random Walk in case of 8 move + stochastic wind.

Sarsa(0) is still much much better, which implies that the agent still learns to reach the end state well, despite the presence of the wind.

## References

[1] Reinforcement Learning by Sutton and Barto (2018)