CS747: ASSIGNMENT 3

SUMRIT GUPTA

170040044

The submission folder contains 4 files –

1. run.sh which is the main shell script used to run the program. The script takes 1 argument, which can be either 'default', 'kings', 'stochastic' for only Sarsa (Task 2,3,4) or 'default_all', 'kings_all', 'stochastic_all' for Task 5.
2. run_and_plot.py which is the entry point of the main program.
3. environments.py which contains the definitions for classes for each type of windy gridworld (i.e. default, with king's moves, and with stochasticity)
4. algos.py which features the implementation of all the algorithms.

The values for $\boldsymbol{\alpha}$ and $\boldsymbol{\varepsilon}$ are kept as 0.5 and 0.1 respectively. The number of episodes are kept 900, and the maximum step limit is set to be 10000. A $\boldsymbol{\varepsilon}$-greedy algorithm is used to select an action from a state to go to the next state. This also ensures that the implementation converges and we don't get stuck in loops following a fixed policy $\pi^* = \arg\max Q$.

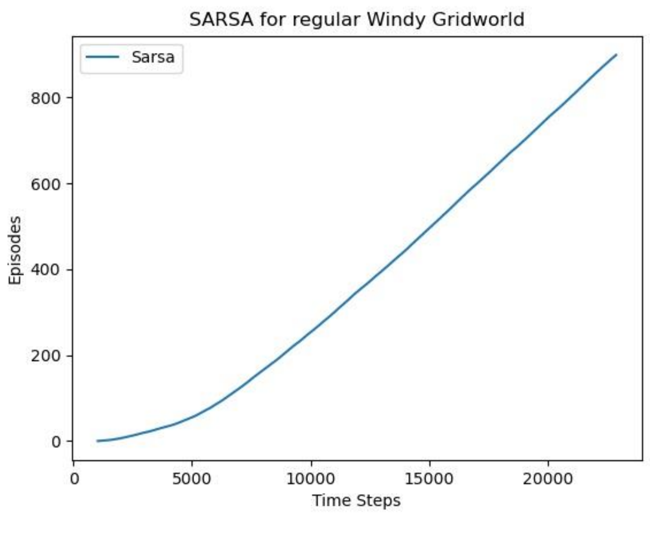
For all the environments $\gamma = 1$.

If the agent is at the border of the gridworld, moving in such a way that it exceeds border limits will force the agent to stay at its original place. This will happen for all gridworld instances (default, kings, stochastic).

The file algos.py has selectAction function which is used to implement -greedy strategy.

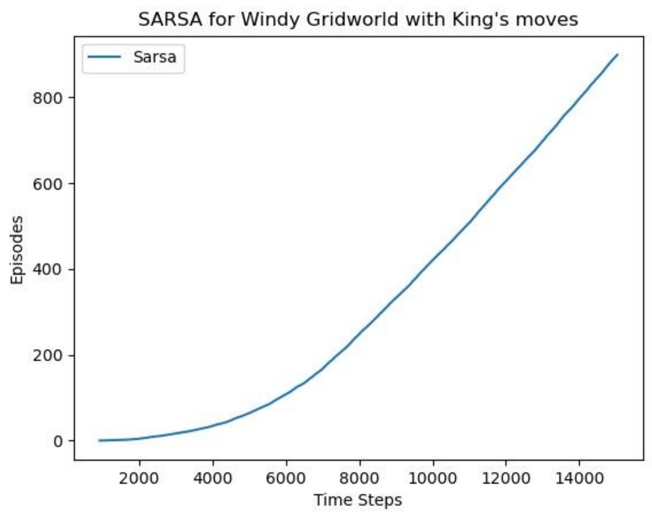
All the plots are generated from results after averaging them over 10 random seeds.

For regular Windy Gridworld



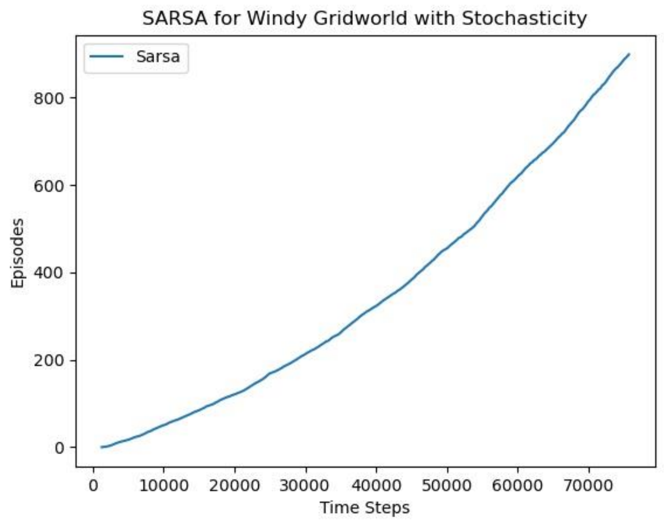
We notice that as the number of episodes increased, the slope of the graph gets steeper and eventually slope becomes constant. This shows that the agent reaches the goal rather quickly after sufficient time steps when Q becomes approximately similar to Q*. This demonstrates that after some time steps, convergence has been achieved corresponding to the shortest possible path.

For Windy Gridworld with King's moves



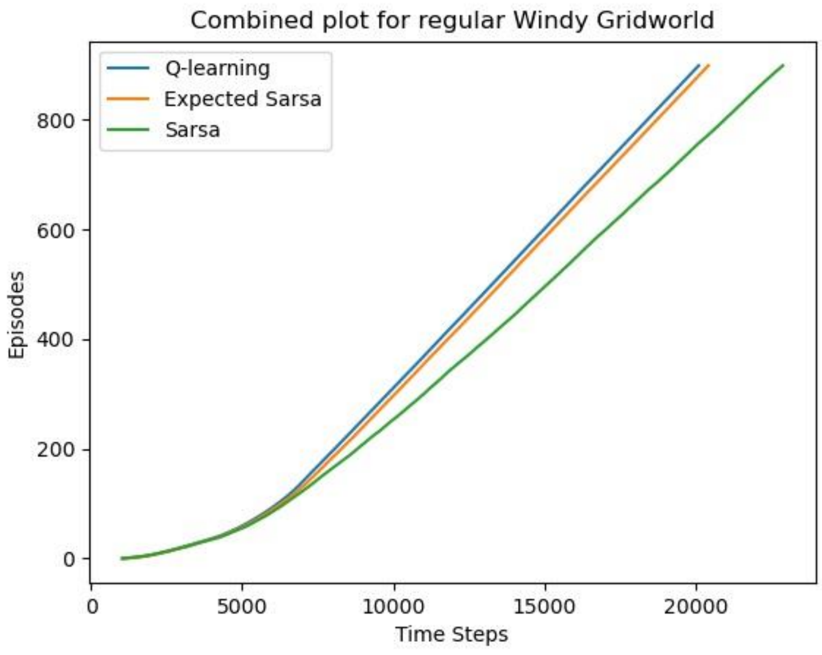
We notice that the graph looks similar to that of the gridworld with 4 moves. But in this case, the agent reaches the goal in lesser number of time steps than in the previous case. This shows that if the agent is allowed more flexibility, it can reach the goal faster than with a limited number of moves.

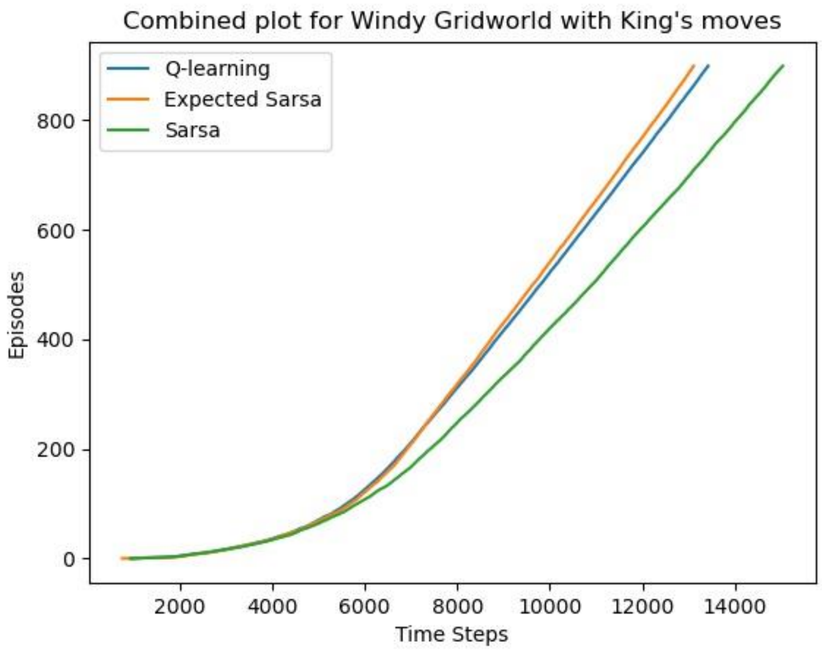
For Windy Gridworld with King's moves and stochastic wind

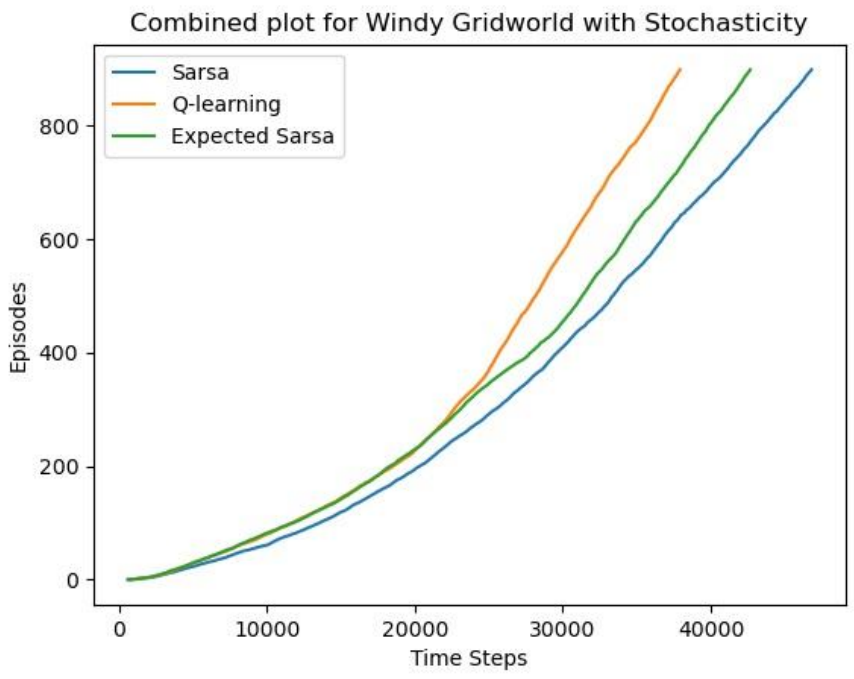


We notice that when winds are kept stochastic, it made reaching the goal more difficult for the agent since the average number of time steps for reaching the goal has increased and hence, it takes longer for the policy to converge. In this graph, it is visible that the policy has not converged yet even after 800 episodes due to high variance.

The graphs below for all the three algorithms are taken as the average over 10 different seed values.







- Here we see that Q-learning and expected Sarsa are quite better than Sarsa and similar to each other.
- In stochastic environments, due to high variance, none of the algorithms provide a good policy as the slopes are not converging but Q-learning is way better than the other two and expected Sarsa performs better than Sarsa.
- So, we can say that more no of episodes might be required to find a good policy.