RL HW EX5:

1. Read and understand Example 6.1. Is there any situation (not necessarily related to this example) where the Monte-Carlo approach might be better than TD? Explain with an example, or explain why not.
    a. The main advantage that TD has over Monte Carlo is that TD methods update their estimates incrementally through bootstrapping where Monte Carlo waits till the end of an episode to update its estimates. However, there are specific situations where the Monte-Carlo approach may actually perform better. When you have a clearly defined episodic task, where rewards are given only at the end or very infrequently, Monte Carlo methods can leverage the full outcome of an episode to update values more accurately. For example, Monte Carlo would be more effective for a game like blackjack, where episodes are relatively short, and the final outcome (win, lose, or draw) is only known at the end of each game, making it ideal for Monte Carlo's full-episode evaluation. That said, if the episodes are very long, the delay in learning until the end of each episode can make Monte Carlo methods inefficient, as updates to estimates happen infrequently, significantly reducing the speed of learning in comparison to that of a TD method.
2. Q-learning vs. SARSA
    a. Why is Q-learning considered an off-policy control method?
        i. Q-learning is considered an off-policy control method because it learns the optimal action-value function Q* by directly approximating it, regardless of the policy being followed. It updates the Q-value by using the maximum Q-value of all possible actions in the next state. This means the updates are made based on the best possible action (according to the learned Q-values), independent of the action that would've been taken by the current policy, making Q-learning an off-policy method.
    b. Suppose action selection is greedy. Is Q-learning then exactly the same algorithm as SARSA? Will they make exactly the same action selections

and weight updates? Explain it briefly.

i. If action selection is greedy, Q-learning and SARSA will make the same action selections because both will always choose the action with the highest Q-value in each state. However, the two algorithms are still not exactly the same. The key difference is in how they update the Q-values. Q-learning updates using the maximum Q-value of the next state (off-policy), while SARSA updates based on the action actually taken by the policy (on-policy). Even though the action selection is identical under a greedy policy, the weight updates may differ because Q-learning updates using the best possible action, whereas SARSA updates based on the chosen action. Thus, the two algorithms may not converge to the same Q-values or solution, especially if the initial Q-values are different.

3. Random-walk task

   a. The specific results shown in the right graph of the random walk example are dependent on the value of the step-size parameter, α. Do you think the conclusions about which algorithm is better would be affected if a wider range of α values were used? Is there a different, fixed value of α at which either algorithm would have performed significantly better than shown? Why or why not?

      i. Conclusions about which algorithm is better could be affected by using a wider range of α values. The performance of both TD and Monte Carlo methods depends on the step-size parameter α, which controls how quickly the algorithms update their value estimates. For TD methods, the accuracy tends to decrease with larger α values, as larger steps can cause the estimates to oscillate around the true values, leading to instability. A smaller α provides more stable updates but at the cost of slower learning. For Monte Carlo, it is less clear if different α values would significantly improve performance, as the graph shows minimal variation based on α. The main limitation of Monte Carlo lies in the fewer value function updates it performs—one per episode. In contrast, TD methods update after every step within an episode. For example, in 100 episodes, TD methods with an average episode length of 6 steps make approximately 600 updates, while

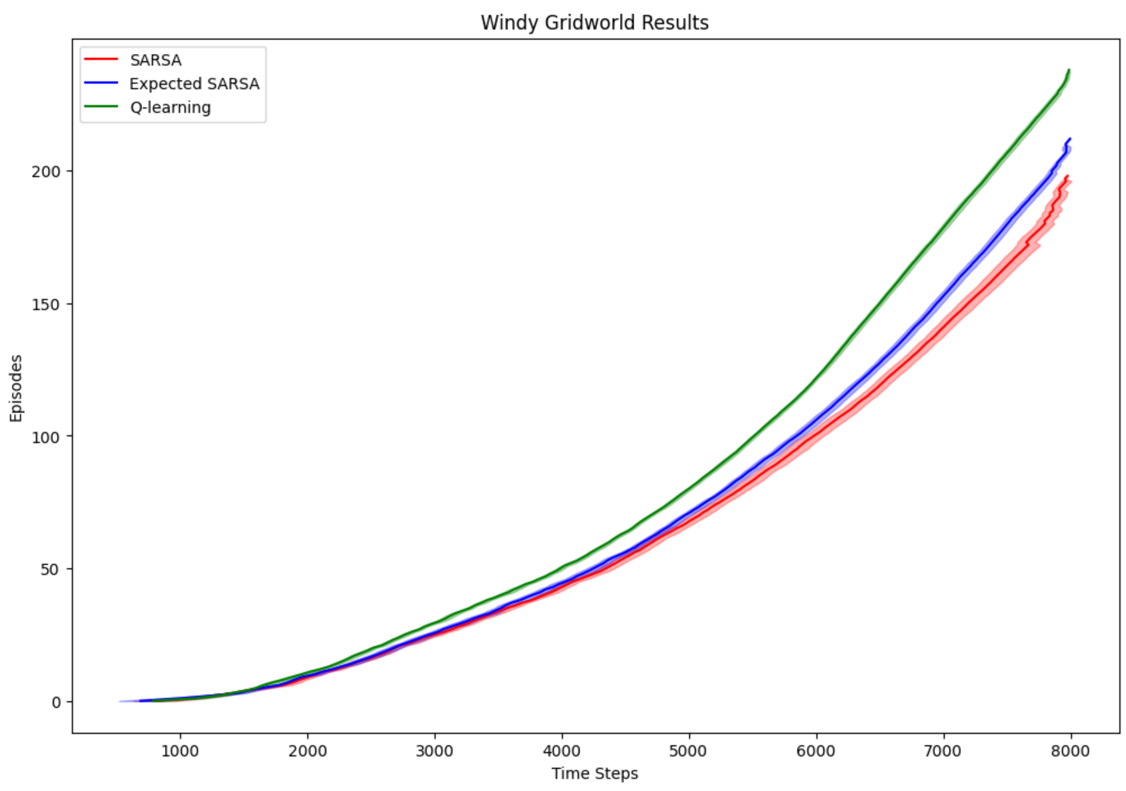
Monte Carlo makes only 100 updates. Therefore, while choosing an optimal α could slightly improve performance for both methods, especially for TD, no value of α can overcome the reduced number of updates in Monte Carlo. This means that the conclusions about the relative performance of TD and Monte Carlo methods would largely remain the same even if a wider range of α values were explored.

b. In the right graph of the random walk example, the RMS error of the TD method seems to go down and then up again, particularly at high α's. What could have caused this? Do you think this always occurs, or might it be a function of how the approximate value function was initialized? Explain your answer briefly.

    i. In the right graph of the random walk example, the increase in RMS error at high αvalues is caused by **instability** from overly large step sizes. When α is too high, the TD method updates too aggressively, causing the value estimates to oscillate and diverge from the true value function, leading to higher error. This behavior may not always occur and could depend on the **initialization** of the value function. If the initial values are close to the true values, the impact of large α might be less significant. However, if the initial values are far off, large α can amplify instability.

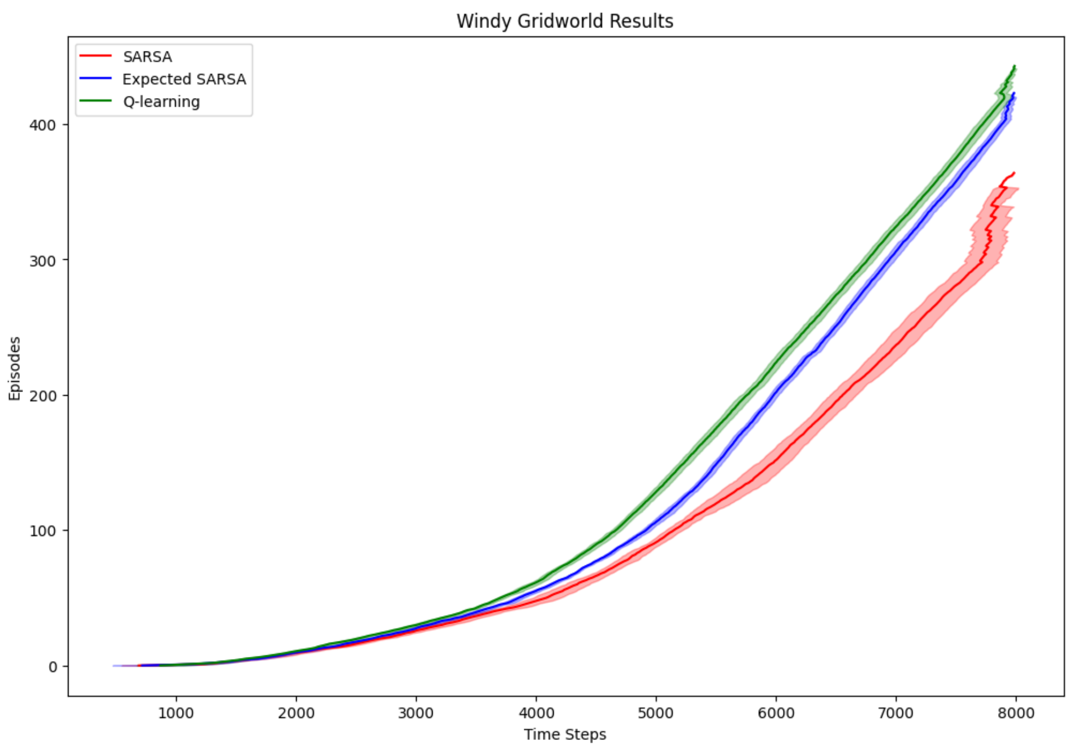
4. Windy gridworld

    a. Code/plot: $\alpha = 0.5, \epsilon = 0.1, \gamma = 1$

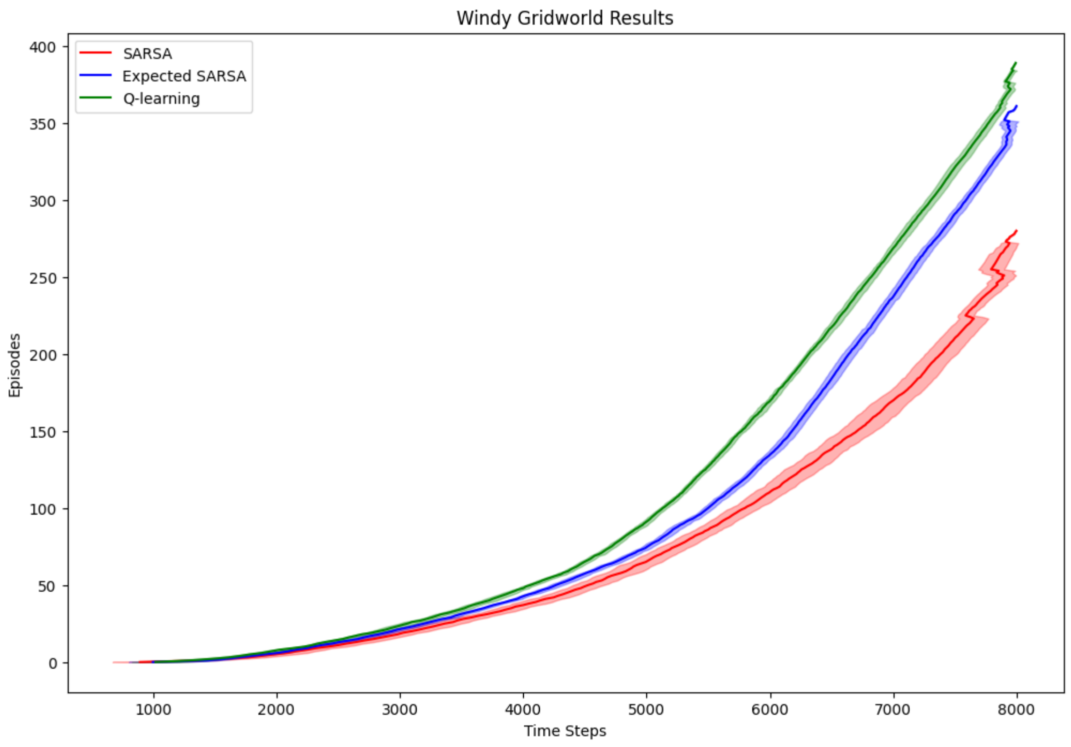


b. Code/plot: $\alpha = 0.5, \epsilon = 0.1, \gamma = 1$

i. King-Moves

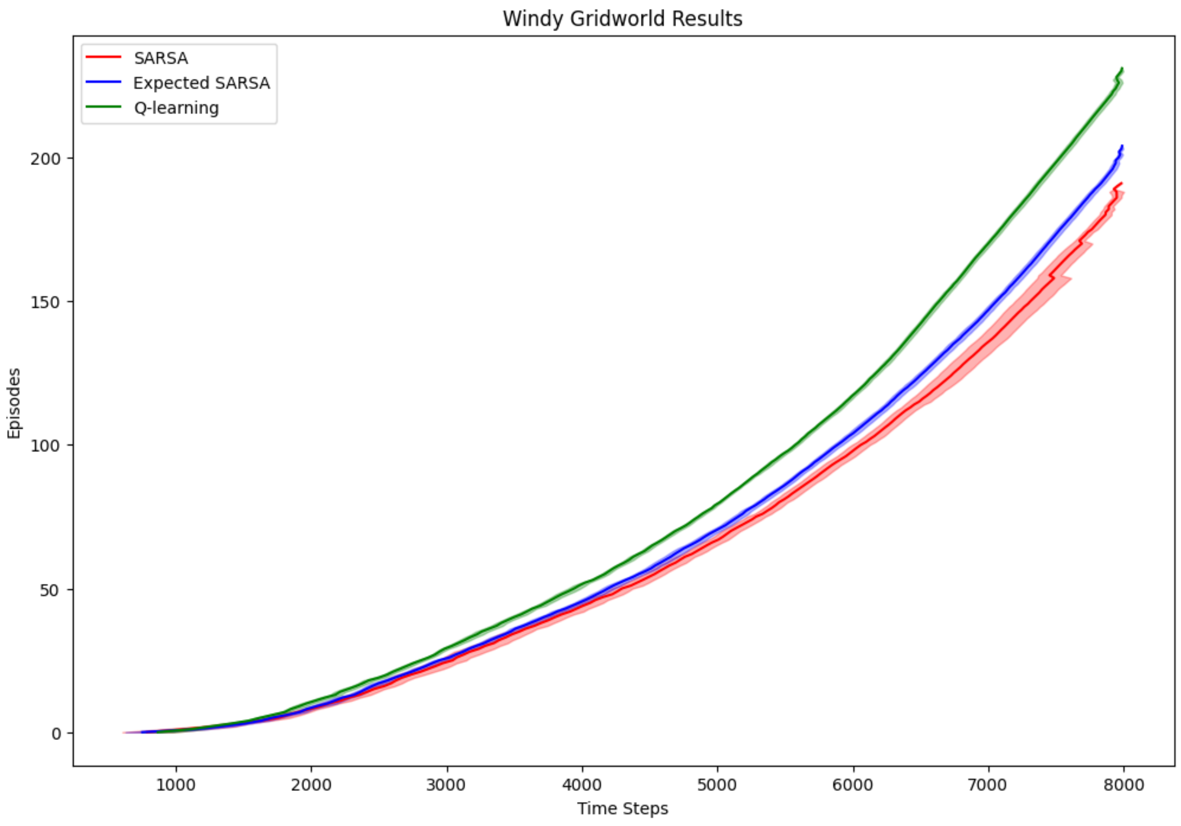


ii. No-Moves + King-Moves

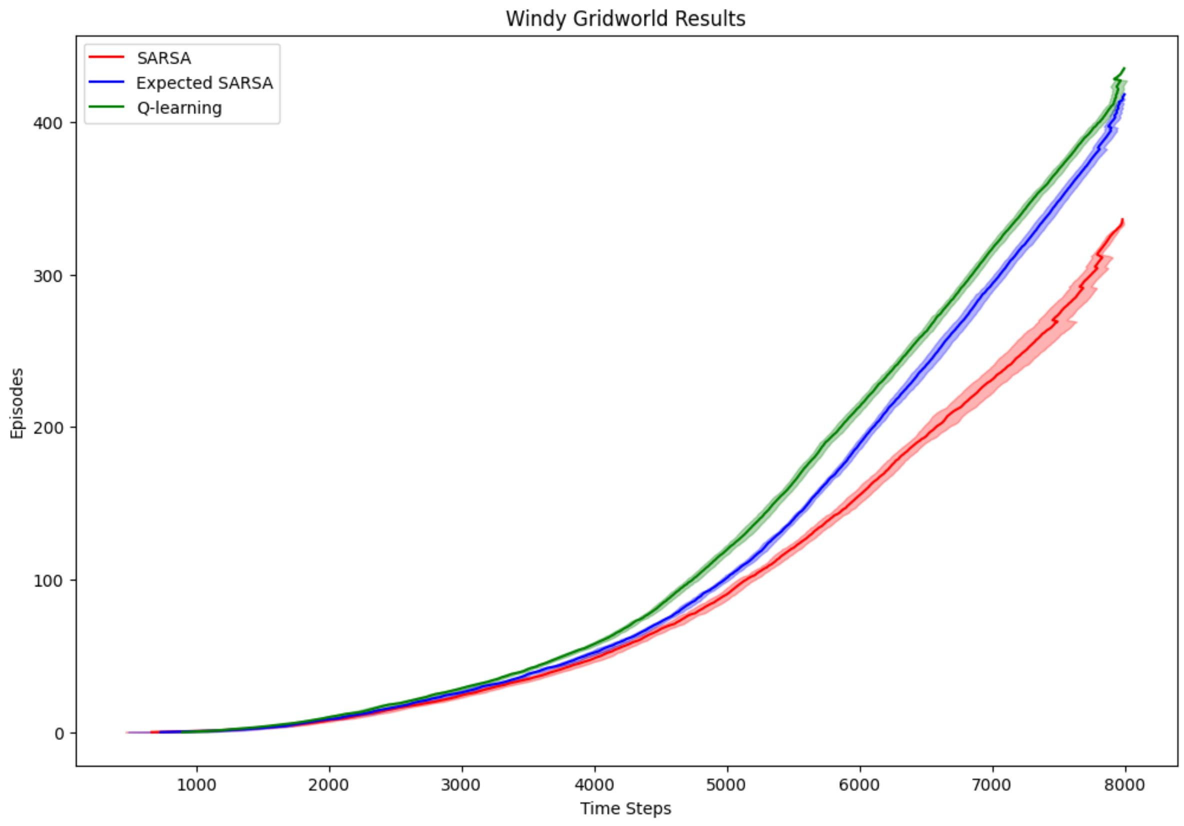


Plots with $\alpha = 0.5, \epsilon = 0.1, \gamma = .9$

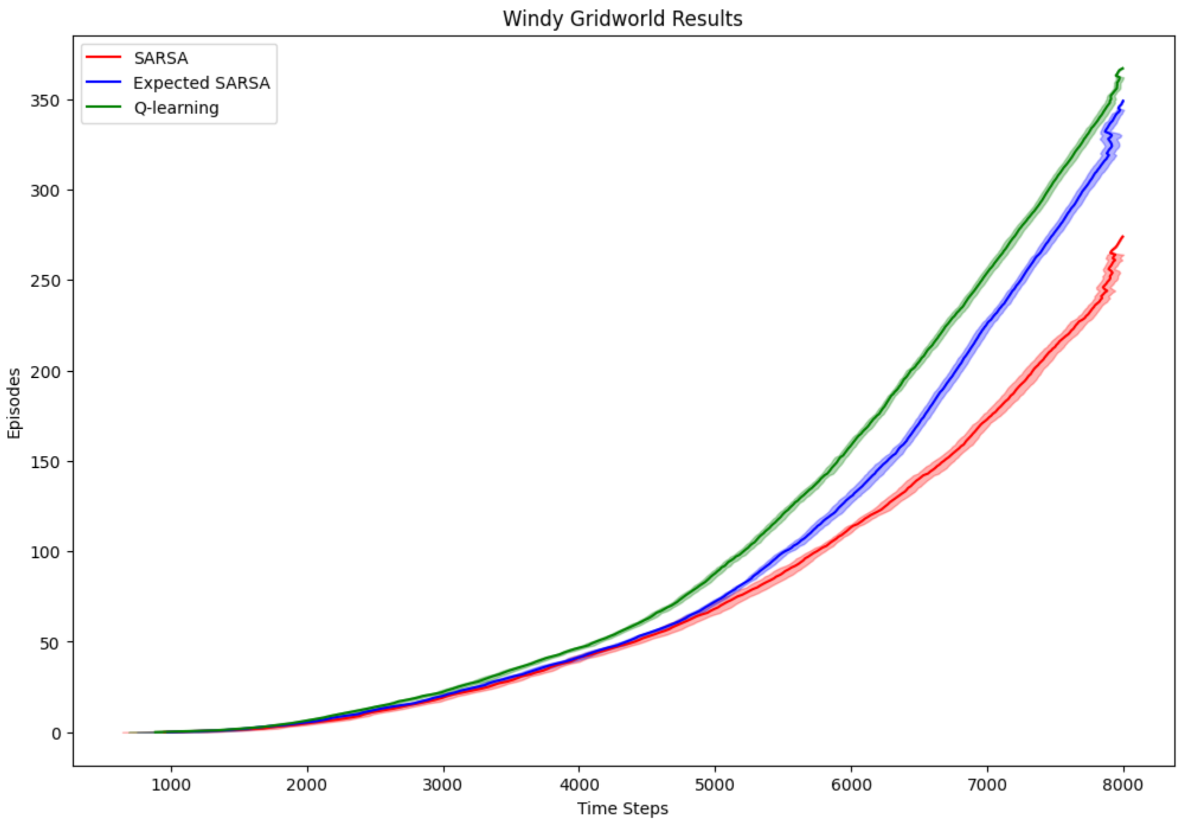
1. Regular:


Windy Gridworld Results
SARSA
Expected SARSA
Q-learning
Episodes
Time Steps
0
50
100
150
200
1000
2000
3000
4000
5000
6000
7000
8000

2. King-Moves



3. No-Moves + King-Moves


Windy Gridworld Results
SARSA
Expected SARSA
Q-learning
Episodes
Time Steps
0
50
100
150
200
250
300
350
1000
2000
3000
4000
5000
6000
7000
8000