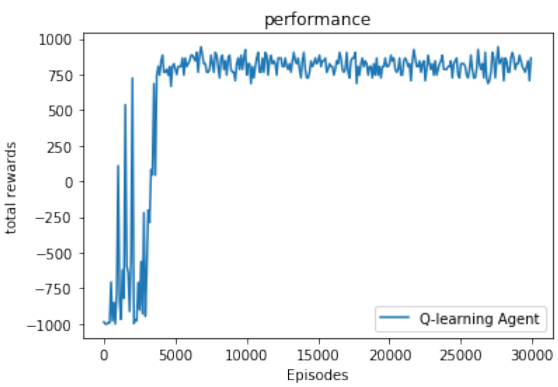
CS 533 Reinforcement Learning
HW 3

Mohamad Hosein Danesh

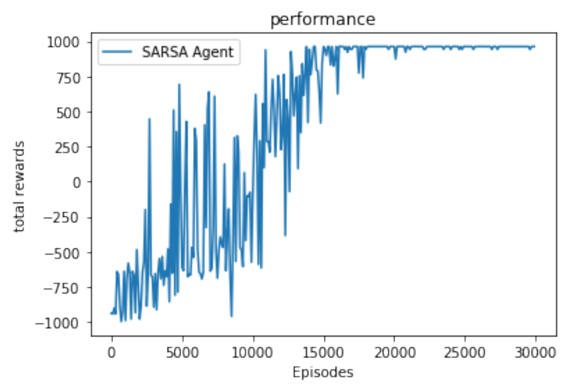
1)

a: Learning Curves (DH):

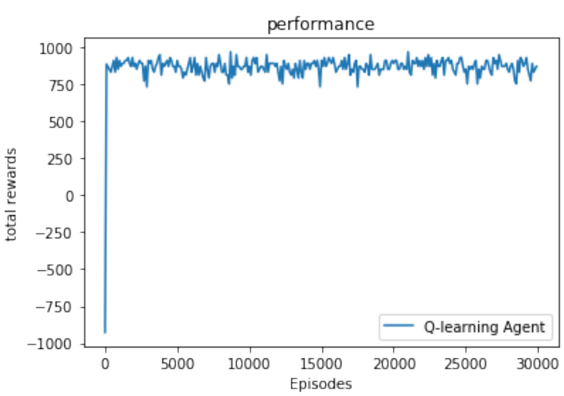
QLearning (epsilon=0.3, lr=0.001)



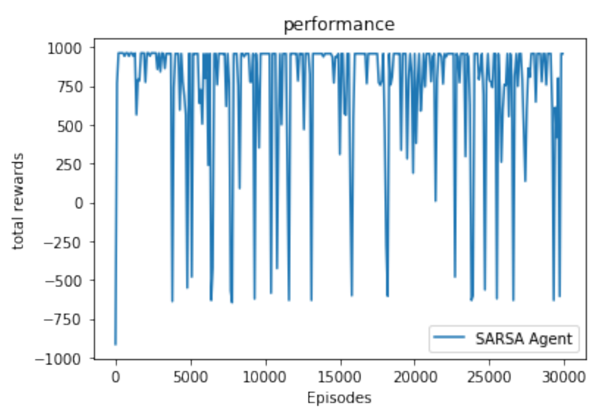
SARSA (epsilon=0.3, lr=0.001)



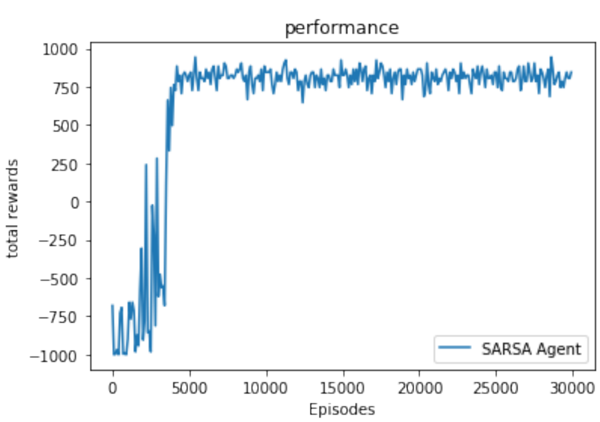
QLearning (epsilon=0.3, lr=0.1)



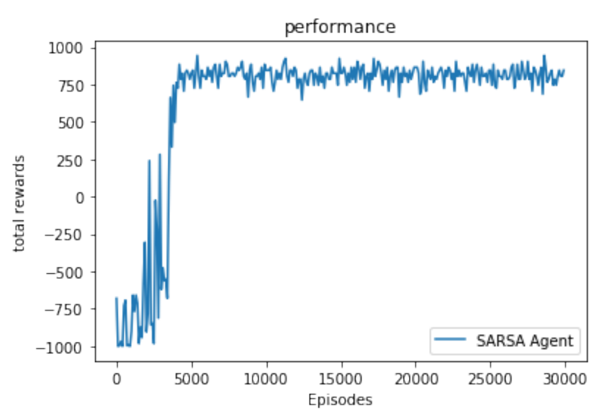
SARSA (epsilon=0.3, lr=0.1)



QLearning (epsilon=0.05, lr=0.001)

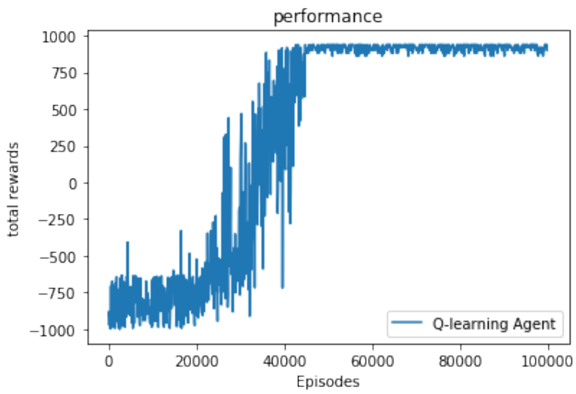


SARSA (epsilon=0.05, lr=0.001)

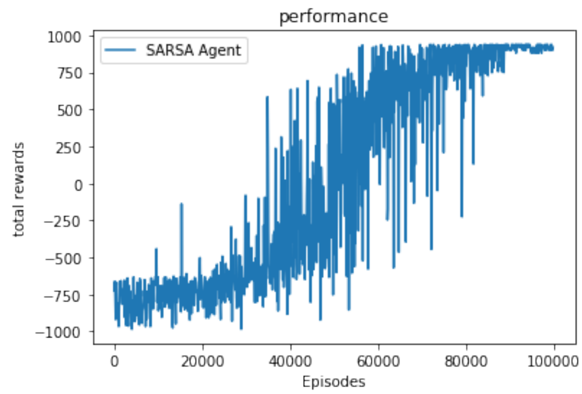


b: Learning Performance (map 16):

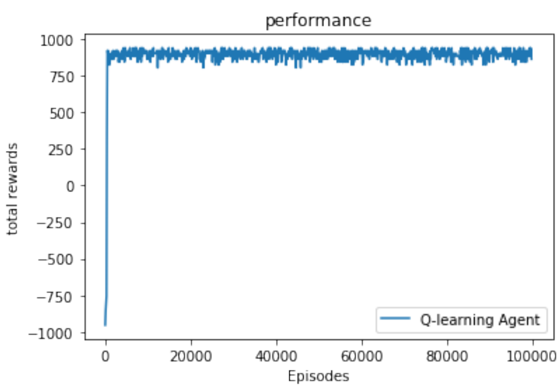
QLearning (epsilon=0.3, lr=0.001)



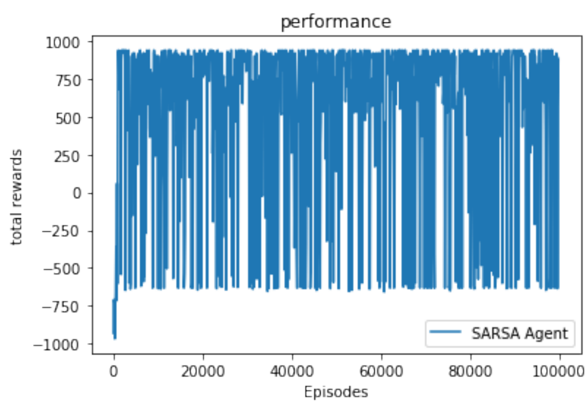
SARSA (epsilon=0.3, lr=0.001)



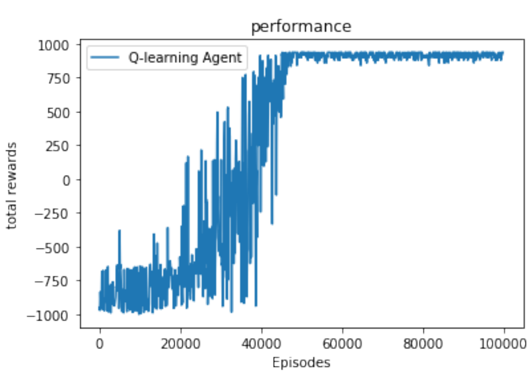
QLearning (epsilon=0.3, lr=0.1)



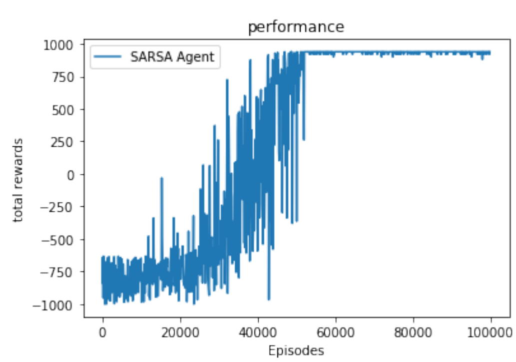
SARSA (epsilon=0.3, lr=0.1)



QLearning (epsilon=0.05, lr=0.001)



SARSA (epsilon=0.05, lr=0.001)



2) When learning rate is high, SARSA is unable to always have the highest total reward and get optimum performance. Therefore, rl=0.001 has the best results.

3) In Q Learning, at higher learning rate has the optimum performance since it converges faster which is because of the taking action with greedy policy.

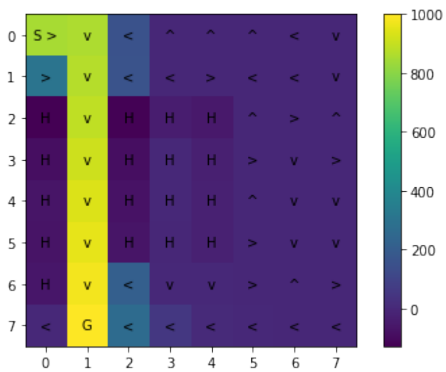
4) SARSA represent optimum performance at lower values of epsilon since it helps to take actions based on the value of states.

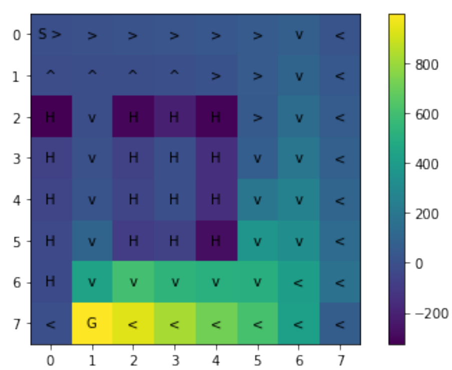
5) Unlike SARSA, Qlearning has better performance at epsilon=0.3 since Q learning takes action based on greedy policy so having higher epsilon lets the agent take enough random actions and explore unseen states and action pairs.

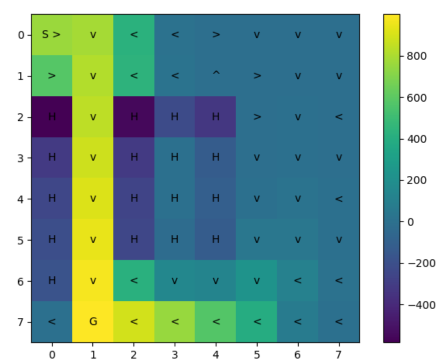
6) SARSA trys to find safest policy and avoids any dangerous state action pairs but Qlearning learns based on the greedy policy and gets the chance to find the quickest way to reach the goal.

7)
Enviornment: DH
Best Q-value and Policy:

QLearning_nonDistributed | SARSA_nonDist | QLearning_Dist(cw=8,ew=4)

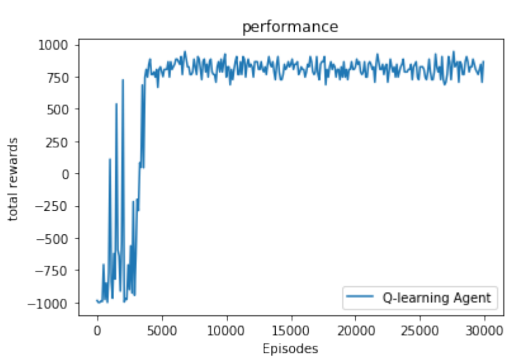


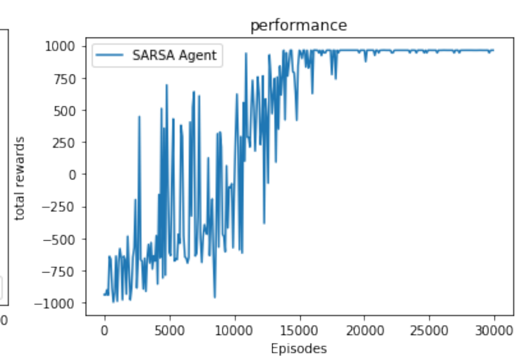


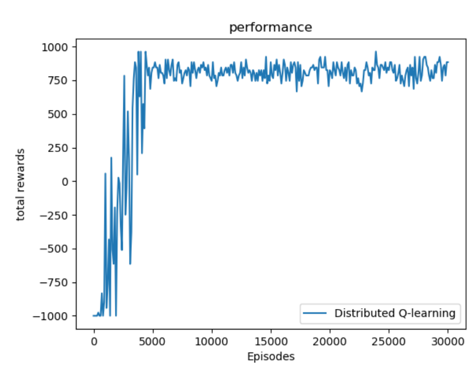


Learning Performance:

QLearning_nonDistributed | SARSA_nonDist | QLearning_Dist(cw=8,ew=4)

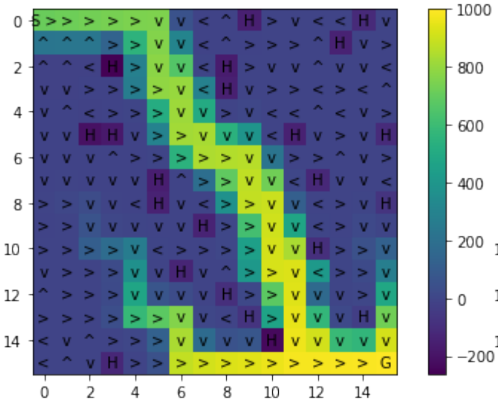


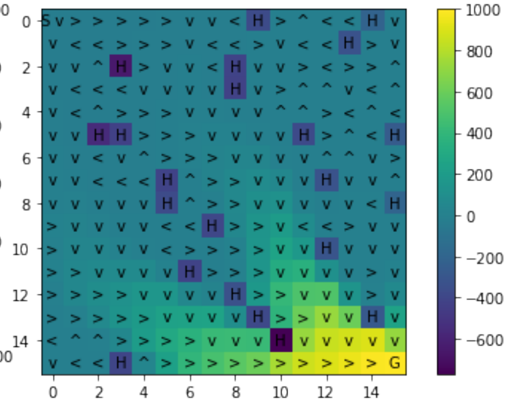


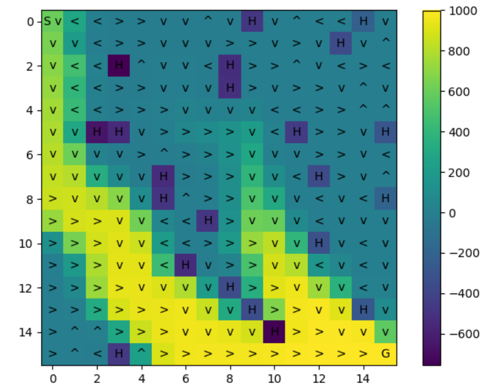


Best Q-value and Policy:

QLearning_nonDistributed SARSA_nonDist QLearning_Dist(cw=16,ew=8)

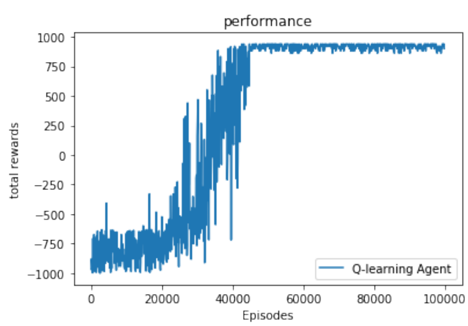


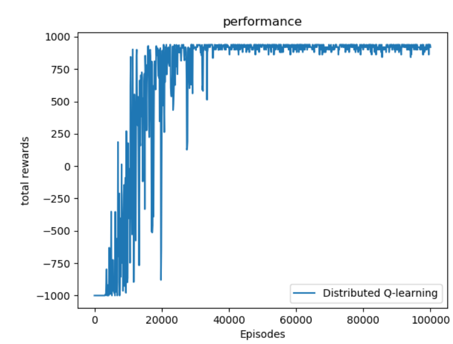




Learning Performance:

QLearning_nonDistributed SARSA_nonDist QLearning_Dist(cw=16,ew=8)





Yes, they produced similar results because the algorithm they used is the same. The only different there is the time they took the agent to learn and perform.

Time Comparison Between Models	With Evaluation Time (DH)
Qlearning_NonDistributed	20.42328453063965
SARSA_NonDistributed	78.29386234283447
Qlearning_Distributed (cw=8,ew=4)	9.260880947113037
Qlearning_Distributed(cw=16,ew=8)	21.29725956916809

Time Comparison Between Models	With Evaluation Time (MAP 16)
Qlearning_NonDistributed	314.4173400402069
SARSA_NonDistributed	508.01805090904236
Qlearning_Distributed (cw=8,ew=4)	98.88091444969177
Qlearning_Distributed(cw=16,ew=8)	45.22370409965515

Increasing the number of workers will decrease the processing time, because there are more workers collecting, learning, and evaluating the agents in parallel. But there is a threshold there that if you increase the number of workers too much, it causes increasing the processing time because of the overhead they create.

9)

Time Comparison Between Models	Without Evaluation Time (DH)
Qlearning_NonDistributed	5.287188768386841
SARSA_NonDistributed	12.68674612045288
Qlearning_Distributed (cw=8,ew=4)	8.59933853149414
Qlearning_Distributed(cw=16,ew=8)	17.177404165267944

Time Comparison Between Models	Without Evaluation Time (MAP 16)
Qlearning_NonDistributed	73.56857204437256
SARSA_NonDistributed	88.60695147514343
Qlearning_Distributed (cw=8,ew=4)	43.40177512168884
Qlearning_Distributed(cw=16,ew=8)	31.30174874015793

In this case, because the time of evaluation is not included in the total time the processing time decreases comparing to the situation where the evaluation time is included, which acts as expected.