CS533

HW3: Distributed Temporal Difference Learning

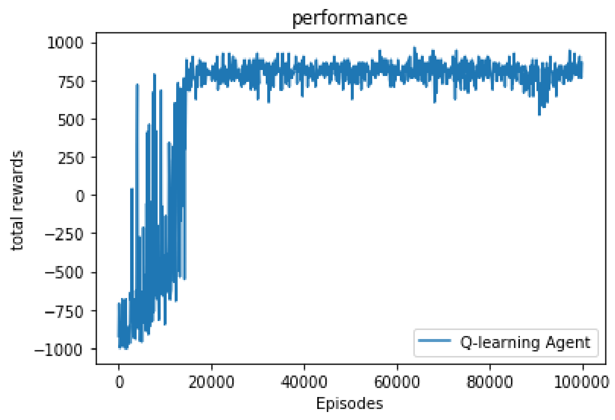
Tenghuan Li; liten@oregonstate.edu
Wenkai Wu; wuwenk@oregonstate.edu

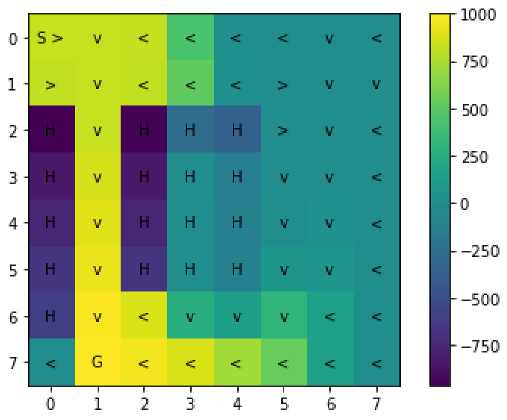
1. Provide the learning curves for the above experiments. Clearly label the curves by the parameters used: **single-core versions:**

map_DH(8*8):

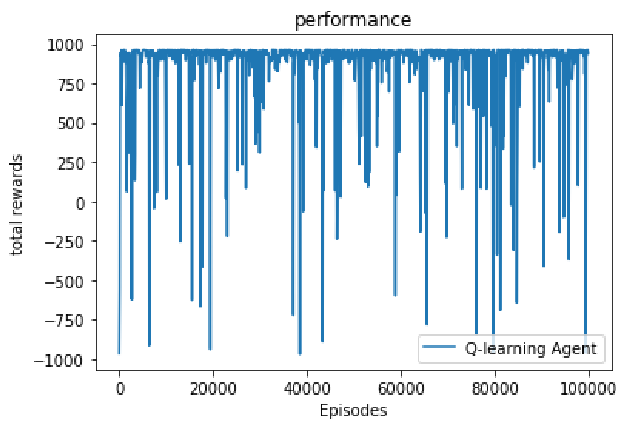
Q-learing:

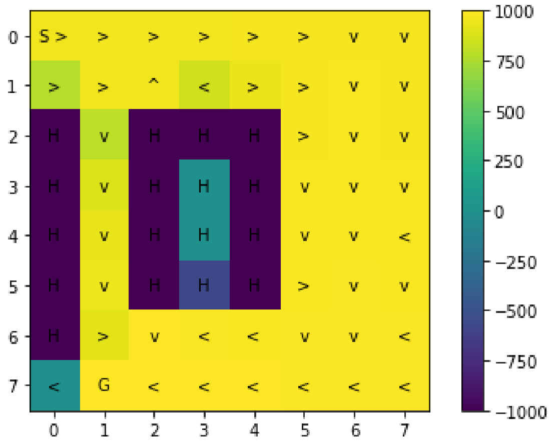
Epsilon = 0.3, learning_rate = 0.001, time = 81.421102523.



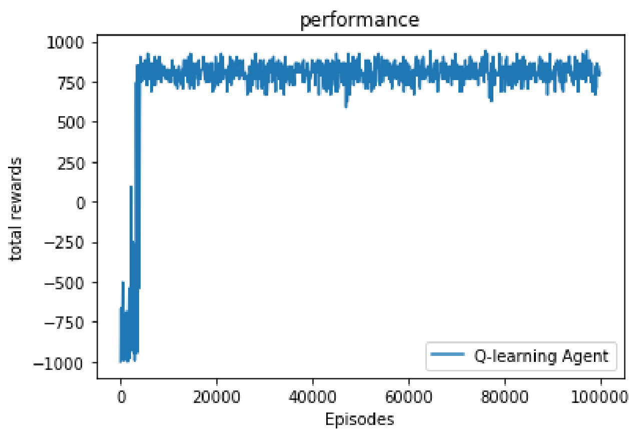


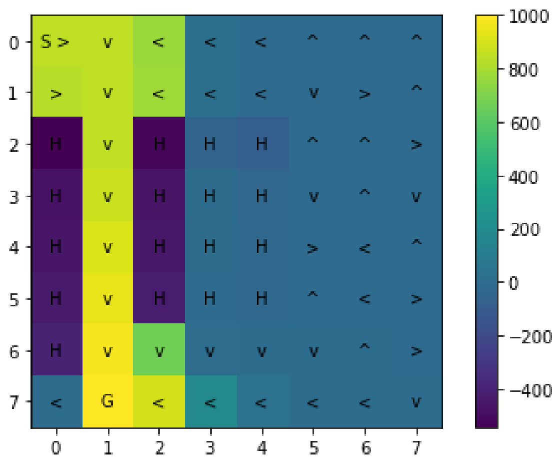
Epsilon = 0.3, learning_rate = 0.1, time = 64.66768217086792.



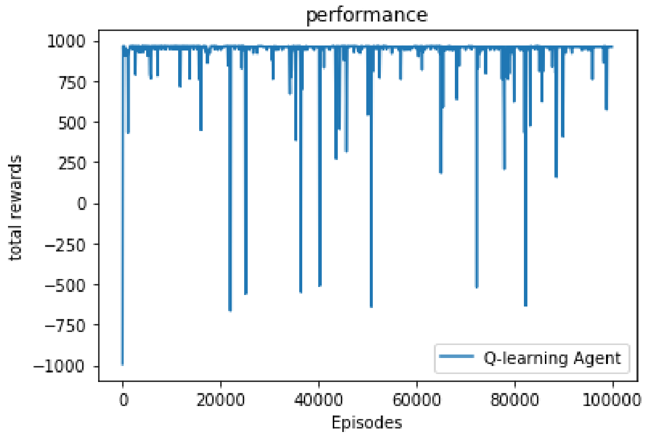


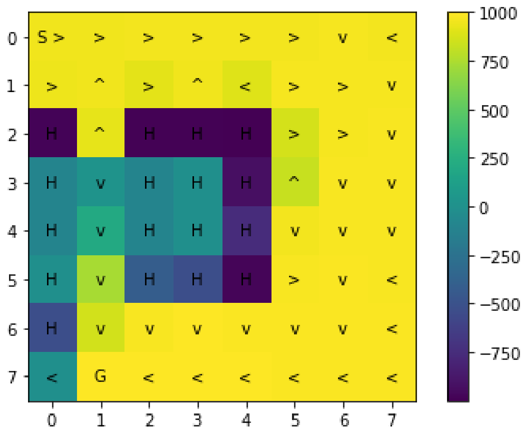
Epsilon = 0.05, learning_rate = 0.001, time = 35.2844493923187256.





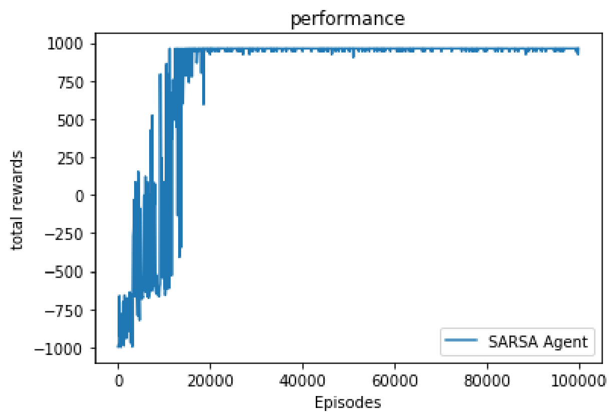
Epsilon = 0.05, learning_rate = 0.1, time = 58.71294069290161.

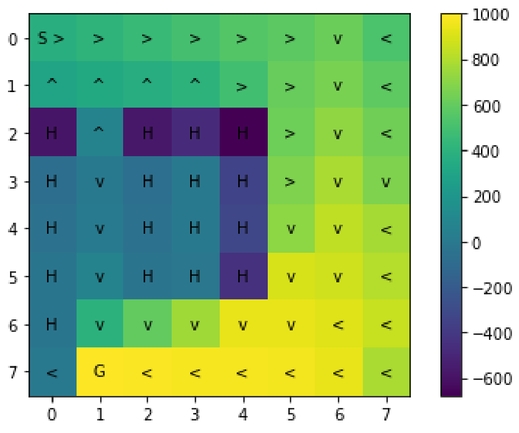




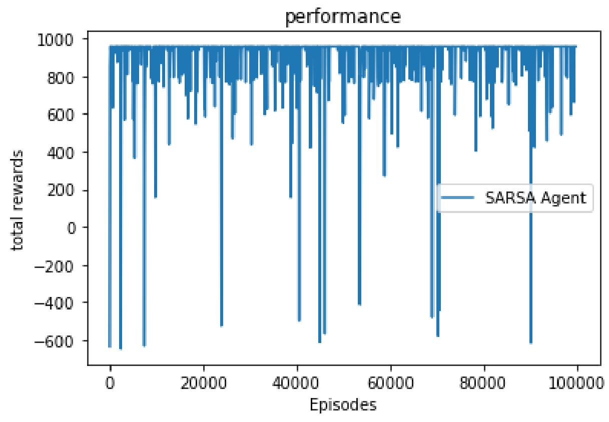
SARSA:

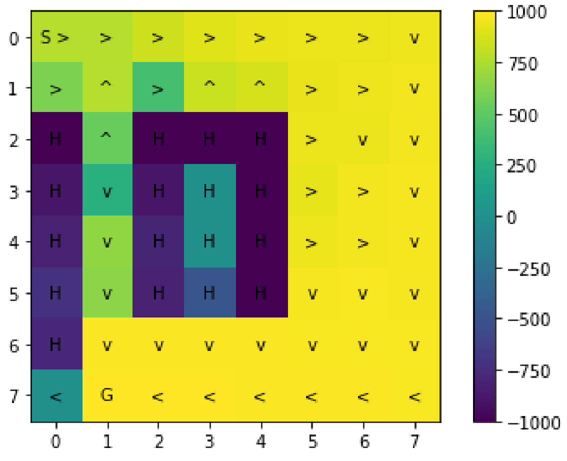
Epsilon = 0.3, learning_rate = 0.001, time = 114.64488005638123.



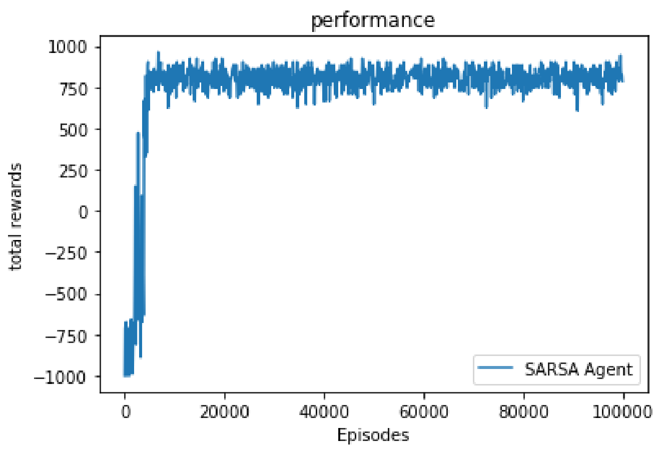


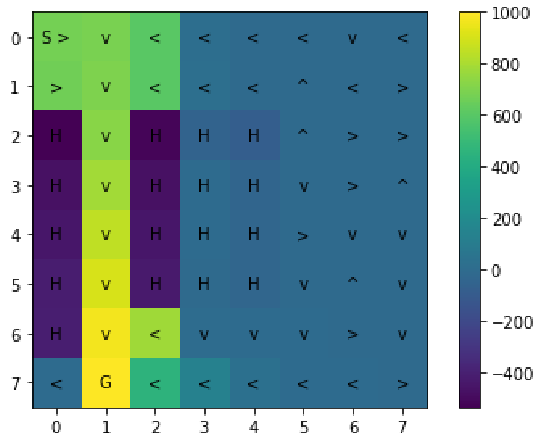
Epsilon = 0.3, learning_rate = 0.1, time =84.18252491950989.



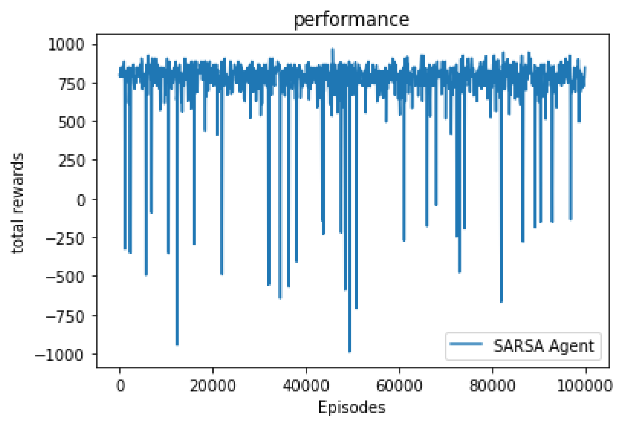


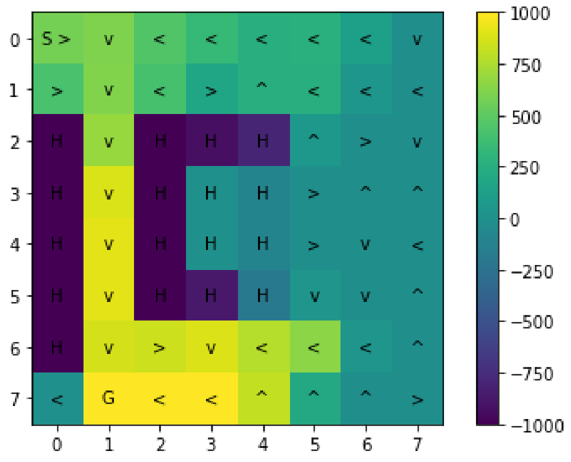
Epsilon = 0.05, learning_rate = 0.001, time =36.5846061706543.





Epsilon = 0.05, learning_rate = 0.1, time = 38.73690938949585.

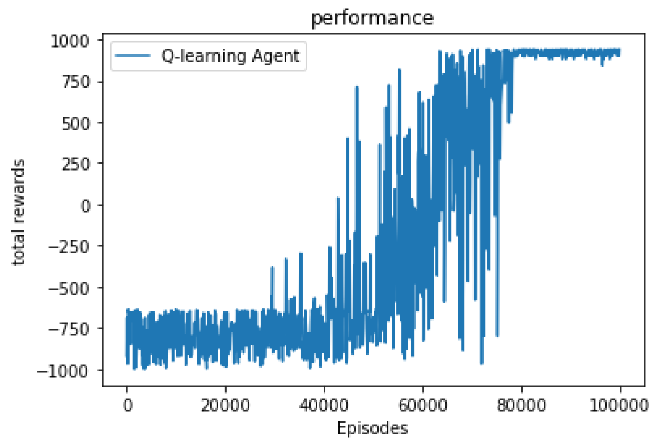


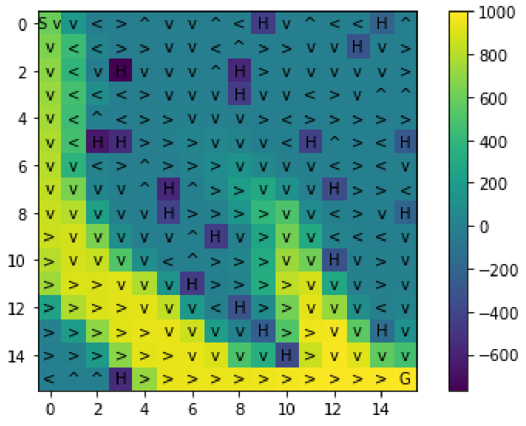


map_16(16*16):

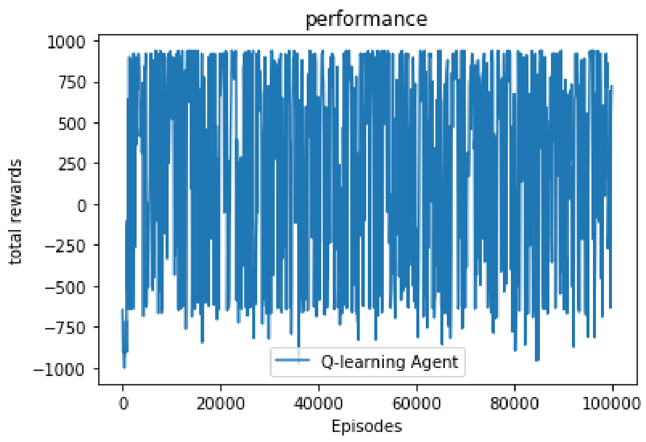
Q-learing:

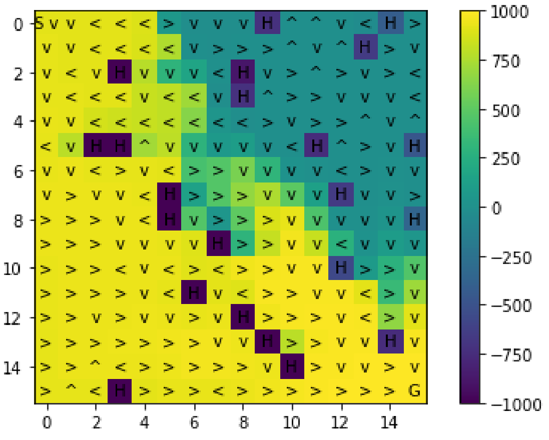
Epsilon = 0.3, learning_rate = 0.001, time = 579.1319191455841.



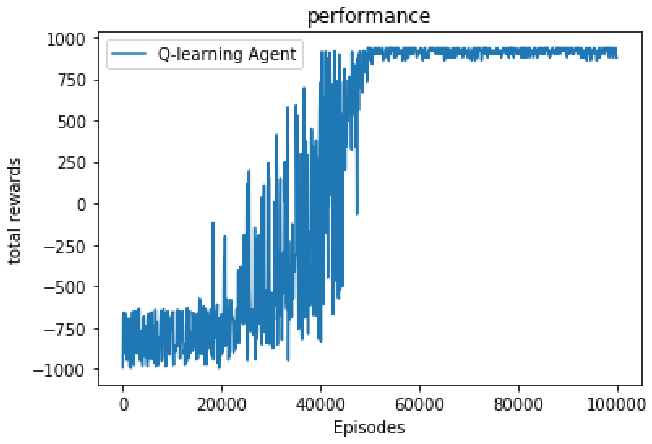


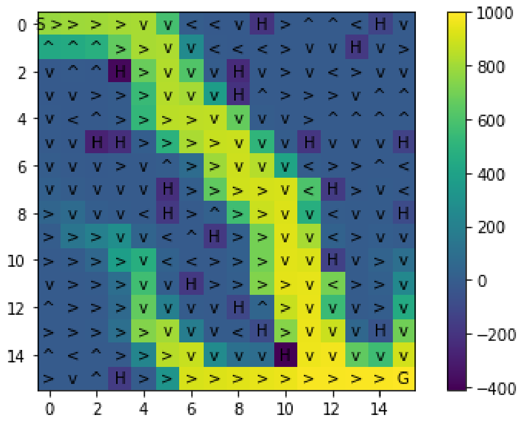
Epsilon = 0.3, learning_rate = 0.1, time = 324.85521173477173.



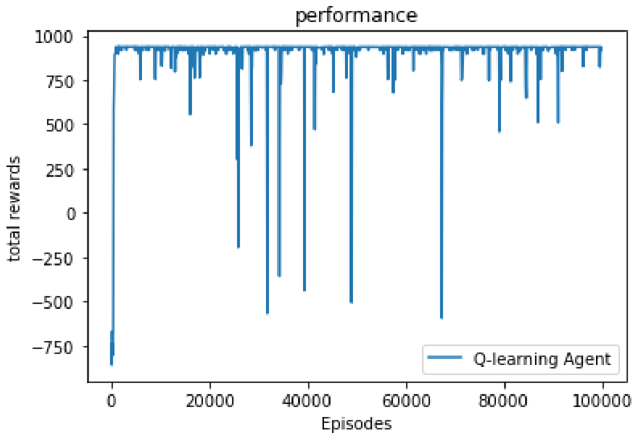


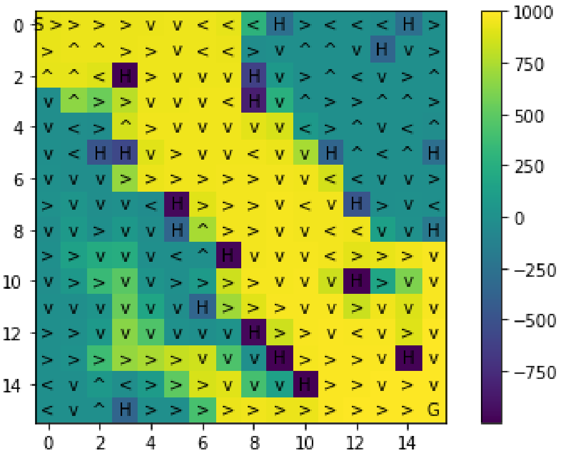
Epsilon = 0.05, learning_rate = 0.001, time = 309.76735639572144.





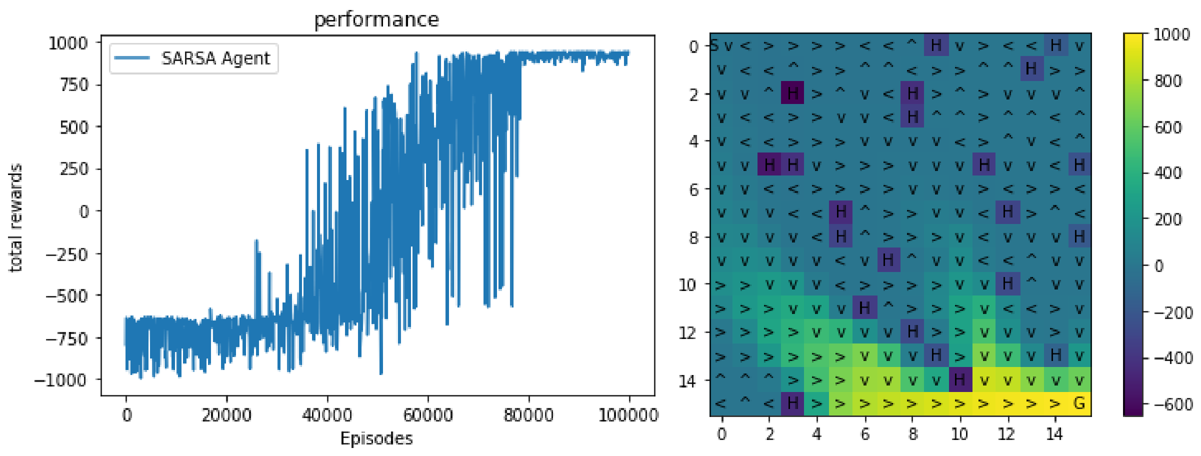
Epsilon = 0.05, learning_rate = 0.1, time = 82.83698606491089.



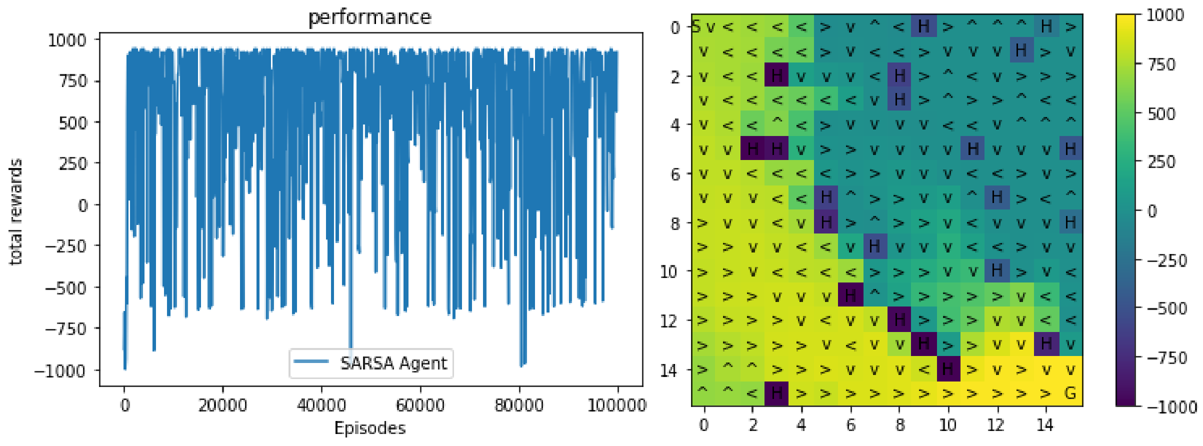


SARSA:

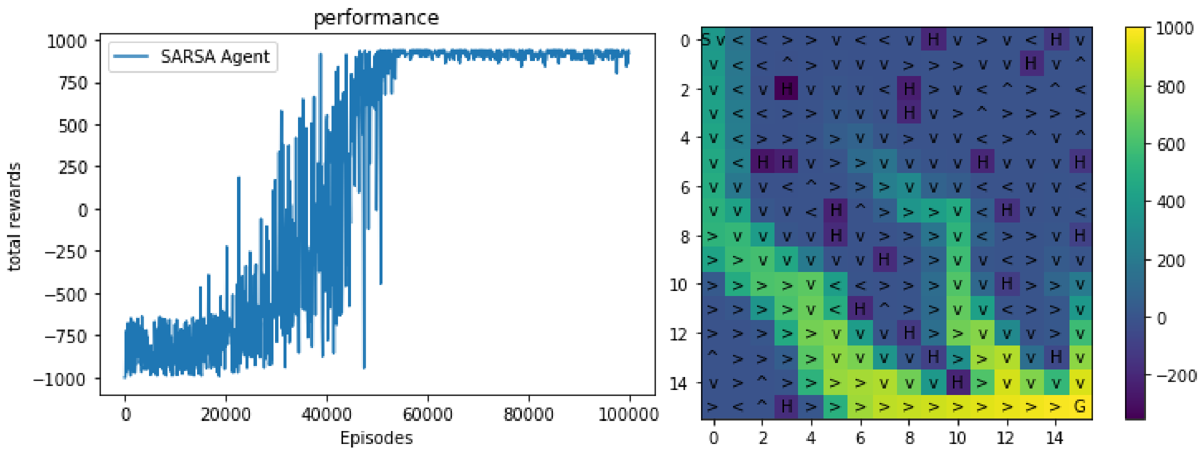
Epsilon = 0.3, learning_rate = 0.001, time =783.7683458328247 .



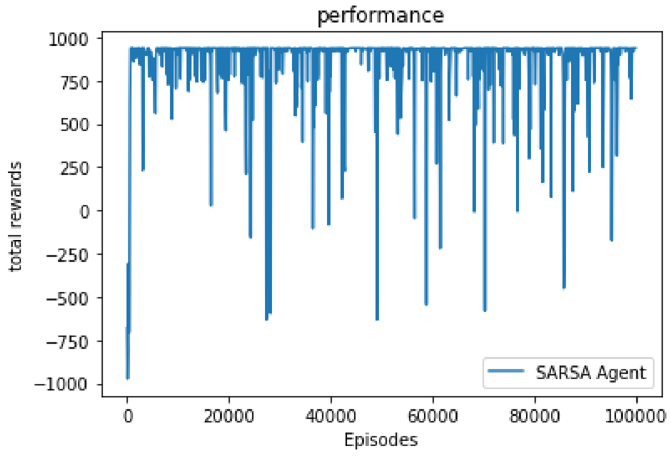
Epsilon = 0.3, learning_rate = 0.1, time = 411.54378151893616.

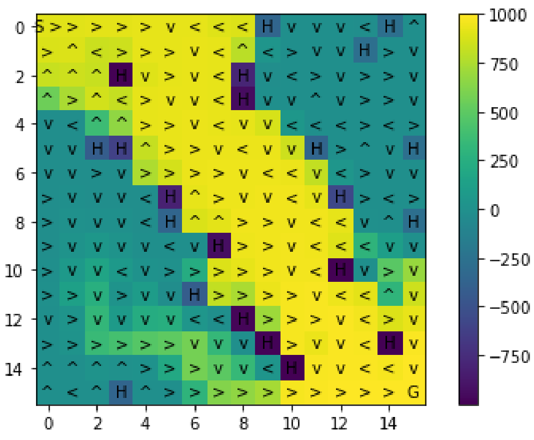


Epsilon = 0.05, learning_rate = 0.001, time = 379.2520155906677.



Epsilon = 0.05, learning_rate = 0.1, time = 184.35492515563965.





2. **Did you observe differences for SARSA when using the two different learning rates? If there were significant differences, what were they and how can you explain them?**
3. **Repeat (2) for Q-Learning.**

In both SARSA and Q-Learning, learning rate determines what the rate is that the old information is override by the new information. Therefore, when the learning rate is 0.1 (closer to 1), the new value we got would have more significant changes. Similarly, when the learning rate is 0.001 (closer to 0), the changes of new value are more conservative.

4. **Did you observe differences for SARSA when using different values of epsilon? If there were significant differences, what were they and how do you explain them?**

Epsilon provides the probability of different choice of next state. In our code, we defined that if epsilon is larger, then the choice of next state is more random. Conversely, if epsilon is smaller, then the choice of next state is more possible decided by greedy policy. Therefore, in our graphic, if epsilon is larger, we can explore more broad area.

5. **Repeat (4) for Q-Learning.**

The influence of epsilon in Q-learning is similar with the one in SARSA. The different part is, due to the Q(s′, a′) in Q-learning is chose by greedy policy, we can see the program showed different path depend on different epsilon obviously (ex. in 16*16 figures).

6. **For the map "Dangerous Hallway" did you observe differences in the policies learned by SARSA and Q-Learning for the two values of epsilon (there should be differences between Q-learning and SARSA for at least one value)? If you observed a difference, give your best explanation for why Q-learning and SARSA found different solutions.**

According to our figure, only when epsilon is 0.3 (higher value) in SARSA is different others. This is because the calculation of new Q(s,a) always depends on the choice of the greedy policy of Q(s′, a′) in Q-learning. And if epsilon is lower, the possibility of using greedy policy is much higher in SARSA.

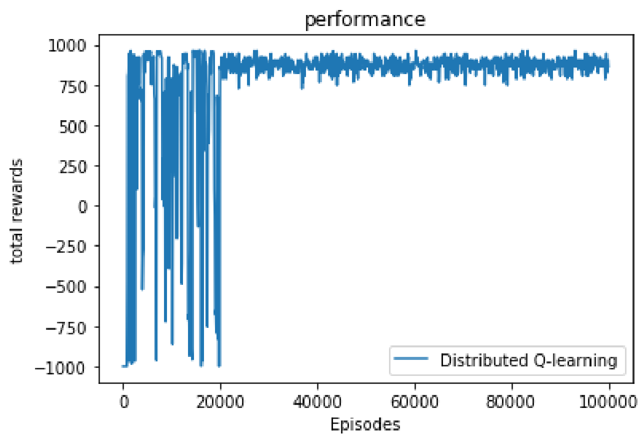
Distributed versions:

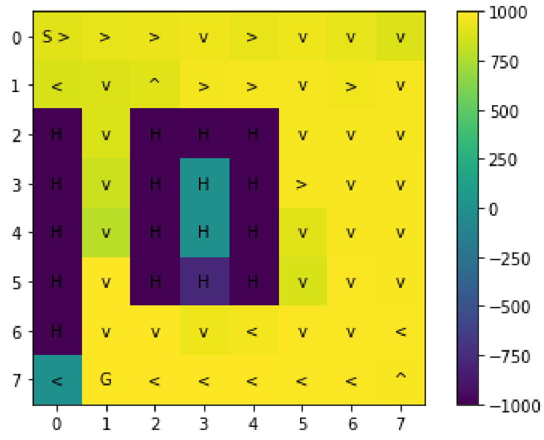
Map_DH:

Epsilon = 0.3, learning rate = 0.1, learning_episodes = 100000, test_interval = 100, batch_size =100.

Worker (2,4):

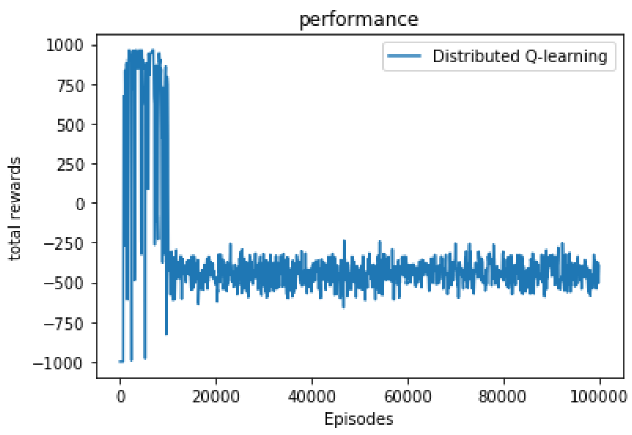
Time = 91.79642534255981

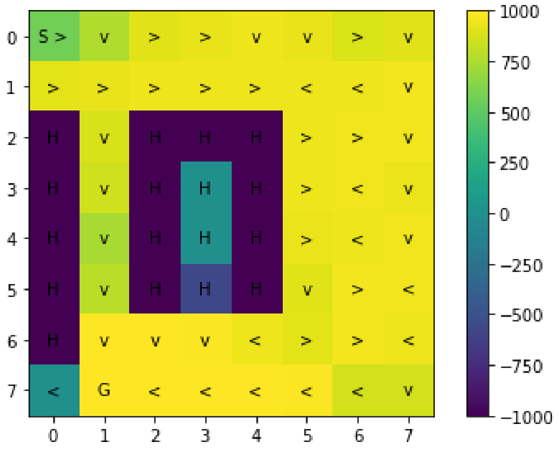




Worker (4,4):

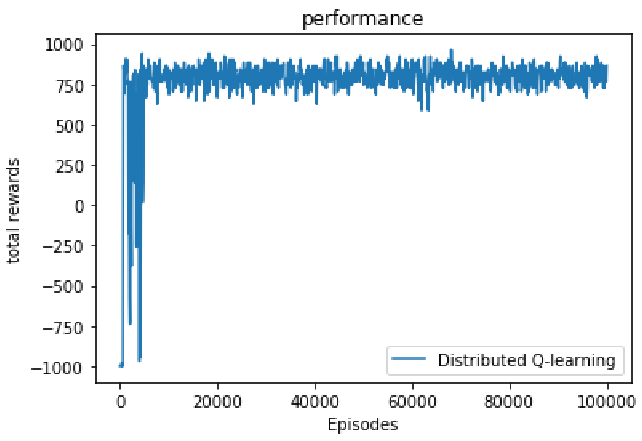
Time = 76.20277404795156

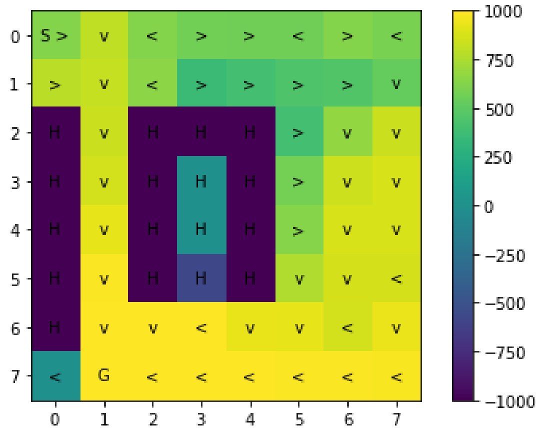




Worker (8,4):

Time = 29.41598916053772



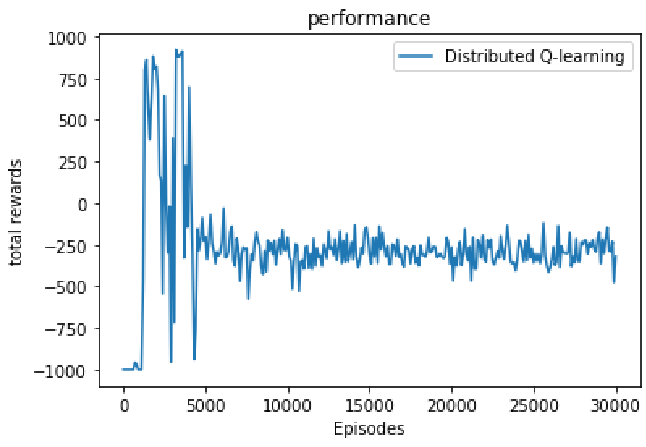


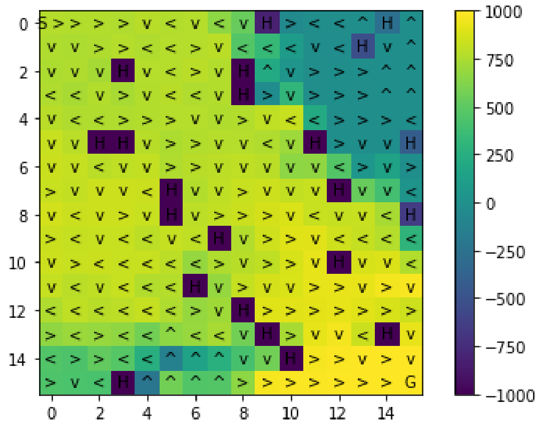
Map_16:

Epsilon = 0.3, learning) rate = 0.1, learning_episodes = 30000, test_interval = 100, batch_size =100.

Worker (2,4):

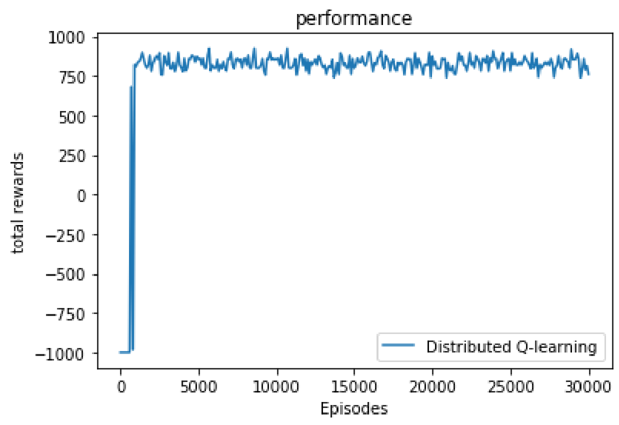
Time = 236.79021406173706

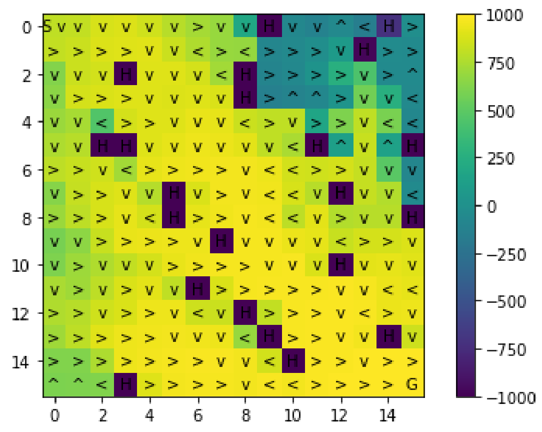




Worker (4,4):

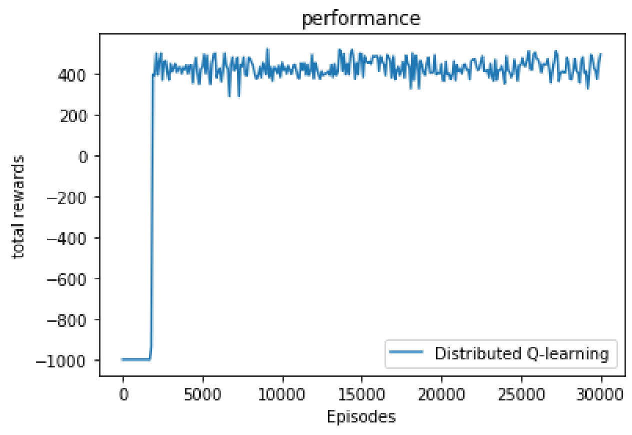
Time = 104.03047108650208

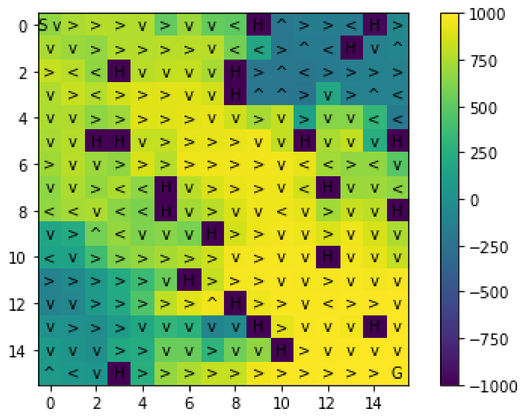




Worker (8,4):

Time = 34.96685528755188



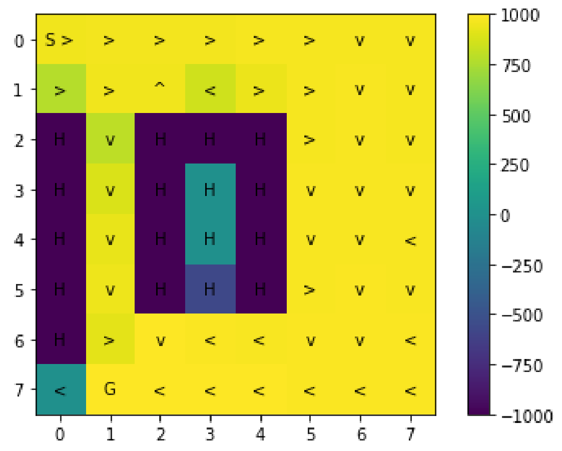


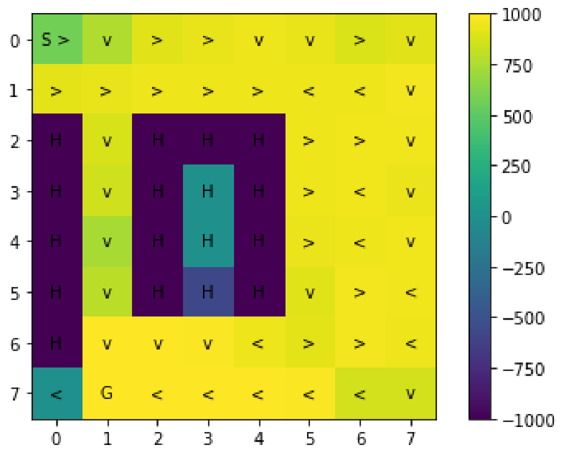
7. Show the value functions learned by the distributed methods for the best policies learned with epsilon equal to 0.3 and compare to those of the single-core method. Run the algorithm for the recommended number of episodes for each of the maps. Did the approaches produce similar results?

a. For map size 8*8:

The Q-learning of epsilon = 0.3, learning_rate = 0.1.

The single-core method gets this result: The distributed methods by worker(4,4) result:

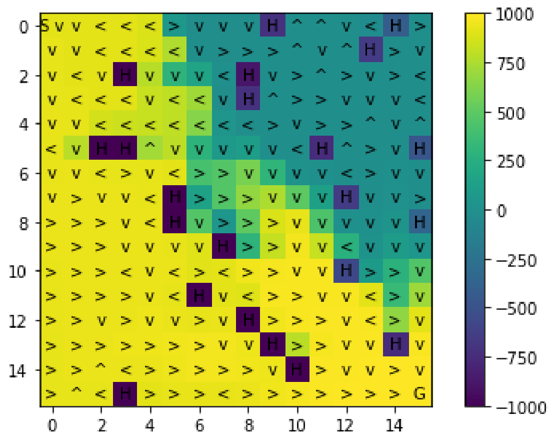


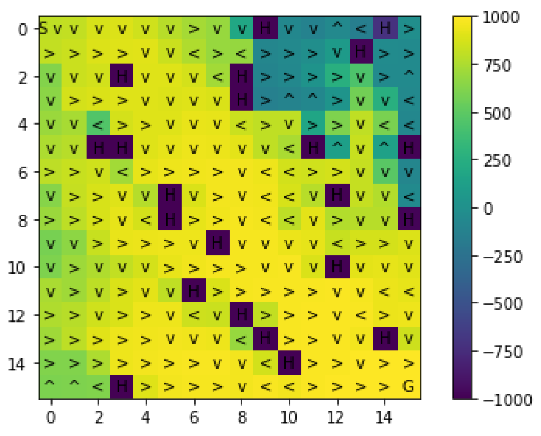


b. For map size 16*16:

The Q-learning of epsilon = 0.3, learning_rate = 0.1.

The single-core method gets this result: The distributed methods by worker(4,4) result:





By comparing the result graphs, we can see that Q-learning on single-core and distributed can produce similar values and policies.

8. Provide and compare the timing results for the single-core and distributed experiments, including the time to do the evaluations during learning. Describe the trends you observe as the number of workers increases.

	Single core	Distributed(2,4)	Distributed(4,4)	Distributed(8,4)
Map 8*8	64.67	91.80	76.20	29.42
Map 16*16	324.86	236.79	104.03	34.96

As the table shows, we can see when the worker is (2,4) the time is close the single core running time. With the increase of collector workers, like (4,4) and (8,4), we can see that the trend of running time go down and become more efficient.

9. Provide and compare the timing results for the single-core and distributed experiments with the evaluation procedure turned off. That is, here you only want to provide timing results for the learning process without any interleaved evaluation. Compare the results with (8).

So, let do_test = False. We get this table as blow:

	Single core	Distributed(2,4)	Distributed(4,4)	Distributed(8,4)
Map 8*8	65.03	31.18	14.51	9.135
Map 16*16	324.86	73.34	35.723	24.62

By comparing the problem 8 the distributed time, we can see when we close the interleaved evaluation the distributed get the faster running time. Because we don't need to do the evaluation, so the overall computational effort is reduced and the effective time complexity is greatly reduced.