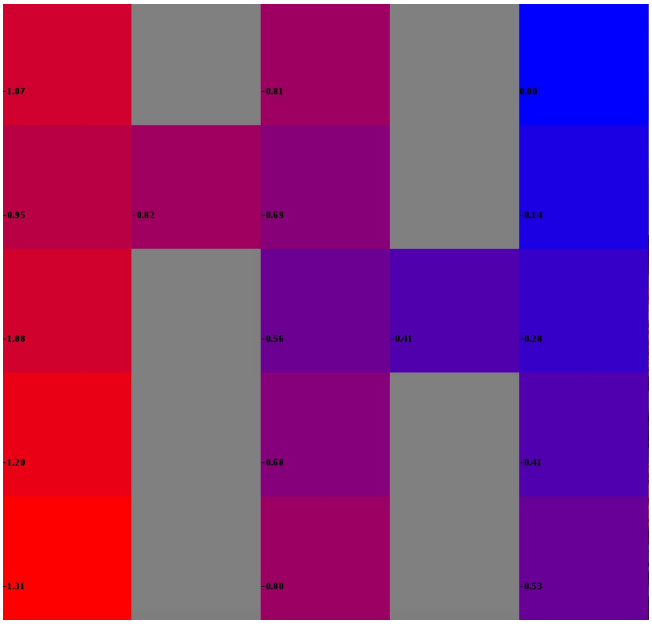
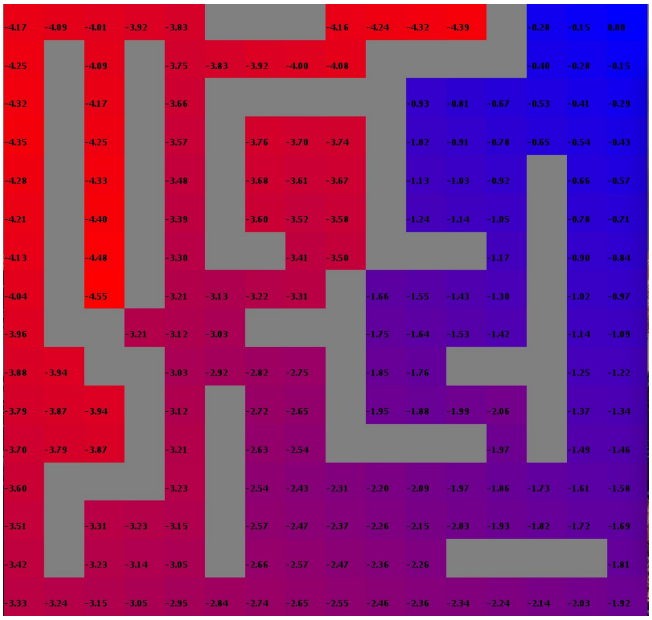
Dantong Zhu
CS 7641
Assignment 4

Markov Decision Processes Write-up

Description of our experiments and the MDPs

In this project, we explore three reinforcement learning algorithms: value iteration (VI), policy iteration (PI), and Q Learning (QL), in which both VI are model-based algorithms whereas QL is model-free. We run each algorithm on two MDPs that are a 5*5 grid and a 16*16 gird. Each MDP is a maze where the bottom left cell is the entrance and the top right cell is the exit of the maze, and in each MDP about 30% of the cells are walls. If a cell in the MDP is not the entrance, the exit, or a wall, we call it a *normal* cell. We set each normal cell to have reward -0.1 and the exist cell to have reward +10. Thus, the optimal solution exists the maze within the smallest number of steps.





Why are the maze MDPs interesting? The above pictures show the result of one of the three algorithms on both MDPs. In each MDO, the grey blocks represent the walls. Ignore the actual values calculated by the algorithm in the pictures for now, and let's take a look at the distribution of the walls cells and the normal cells: Setting every normal cell to have reward -0.1 makes the value of the normal cell only depends on where it is in the maze, instead of the each cell's reward. This makes the MDPs interesting, because this way we can observe for which certain normal cells each algorithm is better at dealing with. For instance, in the small grid, how well can an algorithm figure out what to do when getting stuck at the top or the bottom of the third column? And in the big grid, it might be fairly easy for every algorithm to figure out where to

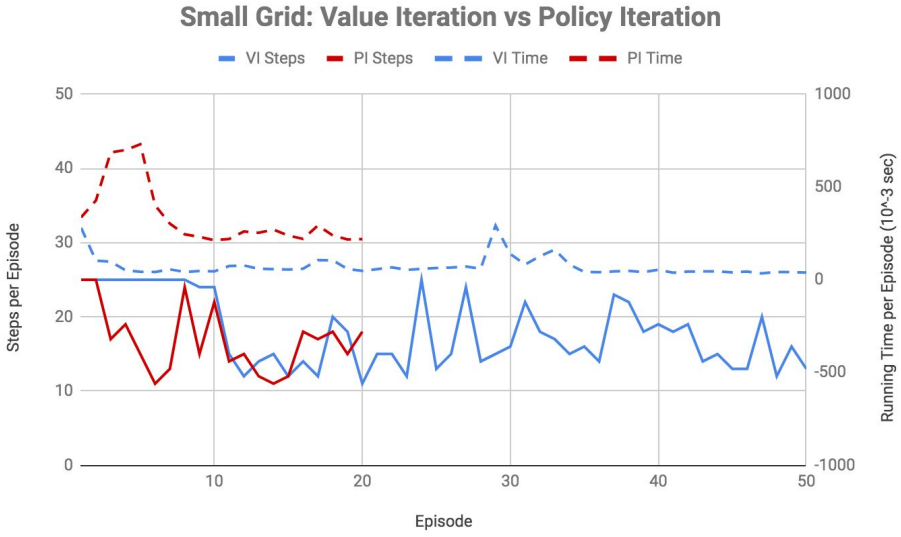
go when on the right side of the maze, but one algorithm might be better than the other one at figuring out what to do when getting stuck at the top of the second column from left or in the top middle part of the maze. The MDPs are interesting because the construction of the walls in them make it possible for us to test the pros and cons for each of the three RL algorithms.

In our experiment, for each algorithm and each MDP, we call each iteration an episode. In each episode, the algorithm gives a protocol to follow, which is the explicit policy in the model-based algorithms or simply the Q values in the Q learning algorithm. At the beginning of the episode, we start at the entrance of the maze and follows the protocol to try to get to the exit. The episode ends if we actually get to the exist or when n*n steps have passed where n is the width of the grid. Note setting the episode end at a certain point is important, since the protocol may just get us stuck at a certain point in the maze. It is important to start over with an updated protocol. In each algorithm and each MDP, we record the number of steps in each episode. The smaller this number the better, as every normal cell has a small negative reward (-0.1) and the exit has a big positive reward (+10).

In the rest of this write-up, we are going to discuss the results obtained from the experiments in detail. We start with discussion value iteration and policy iteration on both MDPs, followed by Q Learning on both MDPs with a comparison between model-based and model-free RL algorithms, and in the end we will explore different learning rates in the Q Learning algorithm.

Value Iteration and Policy Iteration

We first compare results obtained by value iteration and policy iteration on the small grid.



We ran the value iteration for 50 episodes and policy iteration for 20 episodes. For each algorithm and in each episode, we recorded the number of steps used and the running time costed in this episode. We plotted the step numbers in solid lines with vertical axis on the left

and the running times in dashed lines with vertical axis on the right. Note that according our set-up, the maximum number of steps in each episode is 5*5 = 25, and the minimum is 11 which is the number of steps needed in the optimal solution, as one can see in the demonstration of the small grid on page 1 of this write-up. From the result, we make some observations and explain them in the following:

(1) Looking at the two solid lines: They both start at 25 steps, but the red solid line (PI) drops down way earlier than the blue solid line (VI). To be specific, PI goes down to 17 steps in episode 3, whereas VI goes down to 24 steps in episode 9 and down to 15 in episode 11. This shows that PI requires much fewer iterations to figure out a policy/path to get to the exit of the maze with in 25 steps than VI.

(2) However, we cannot just conclude PI is just more efficient an algorithm than VI. This is because, as looking at the dashed running time curves, we observe that in every iteration, PI needs much more time to compute the model/policy for the next iteration. In fact, as PI finds out the way to get out of the maze in 17 steps in episode 3, it has costed total time 1.449 seconds, whereas VI finds the way to get out in 15 steps in episode 11 with total time cost 0.881 second. We try to explain this phenomenon from the construction of the two algorithms as follows: In the VI algorithm when calculating the model from episode t-1 to episode t, the computer simply needs to compute the estimated value function for each state s via the following equation:

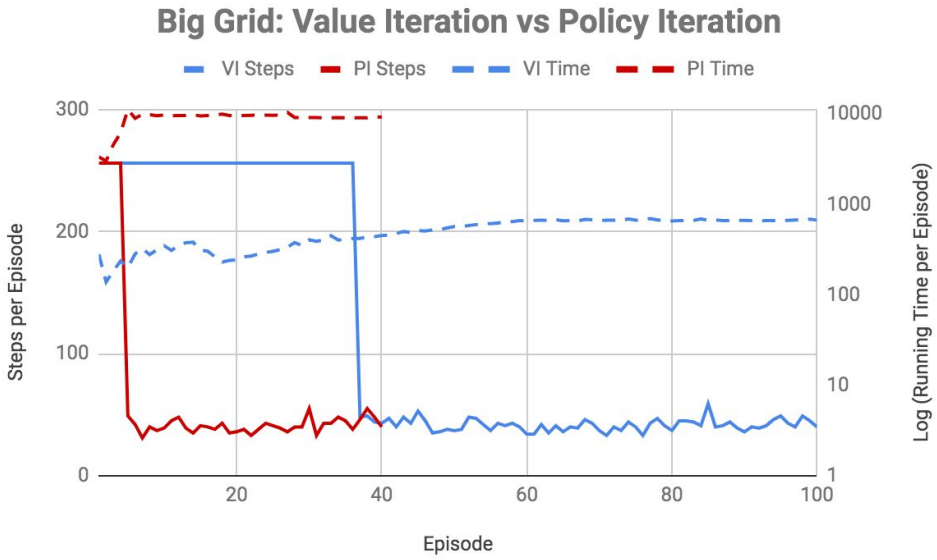
$$V^t(s) = R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s') \cdot V^{t-1}(s').$$

As the $V^{t-1}(s')$ values and the probabilities are known, the computation just focuses on finding the max in the second part of the equation, which is fast. On the other hand, in the PI algorithm, the computer has to solve a linear system with the same number of equations and variables. As solving a linear system requires more computation time for computer, each PI episode is relatively slow.

(3) One other observation is that we see the red dashed line goes up a lot when the red solid line drops. This is saying that the time cost in the episode when PI finds a better policy is bigger than the time costs in previous episodes when the PI cannot get out of the maze. However, a similar phenomenon is not observed for VI. We still try to explain it by the construction of the two algorithms: as PI suddenly finds a way out of the maze, more states become ``active'' since before some states may have not been touched. Even though in each episode, the number of equations and variables to solve for PI stay the same, the activation of more states in the maze probably suddenly increases the rank of the linear system a lot, which then makes the computation time of solving the system much longer than before. On the other hand, to compute the model for the next episode VI always just need to take some maximum, which does not required as much time.

We then run VI and PI on the big grid. By the construction of this MDP and the experiment, the maximum number of steps per episode is 16*16 = 256, and the minimum is 31 since one can walk from the entrance to the very bottom right cell first and then go all the way up to the goal

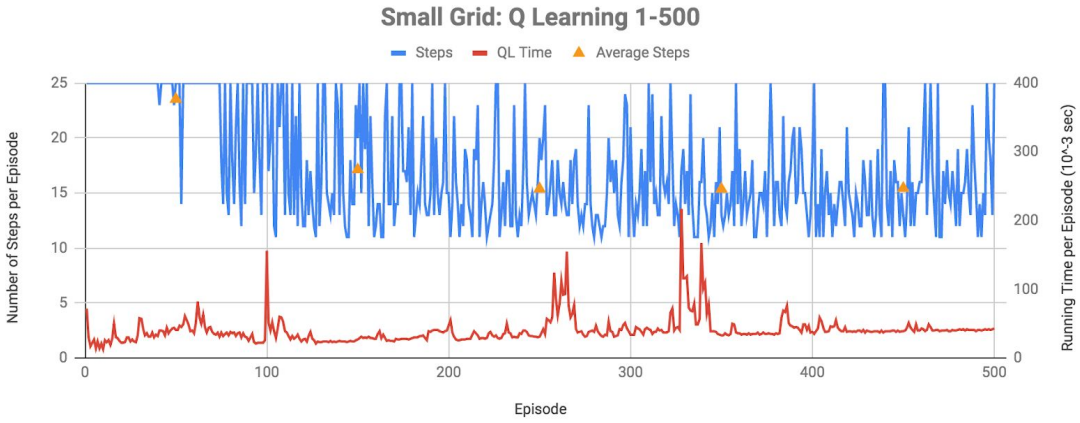
with no obstacle on the way. In our experiment, we run VI for 100 episodes and PI for 40 episodes. We obtain a similar plot in the following.



The observations that we made for the small grid are still true in this plot for the big grid: VI starts to converge in episode 37 whereas PI starts to converge in episode 5 and gets an optimal solution of 31 steps in episode 7. VI takes total 11.742 seconds when it first gets to 48 steps in episode 37 and 31.784 seconds to get an optimal solution in episode 71; whereas when PI finds an optimal policy in episode 7 it used up 47.385 seconds already.

The observations about the running time curves stay the same too. But this time note that we have to take log for the running time curves, as PI requires way too much time in each episode. This difference of running time in the large MDP is way more than that of the small MDP. We believe this is still because of the amount of computation needed for solving a linear system for PI. For a linear system containing m variables and m equations requires running time in some polynomial order of m, say m^k. Note m is our scenario is exactly the number of states, which jumps from 25 in the small MDP to 256 in the large MDP. This means the theoretical difference of the running time in the two MDPs would be (256/25)^k, whereas the difference for VI would just be 256/25. This explains why PI in large MDP requires so much computation time. And we conclude that in an MDP with a big number of states, VI performs better than PI.

Q Learning

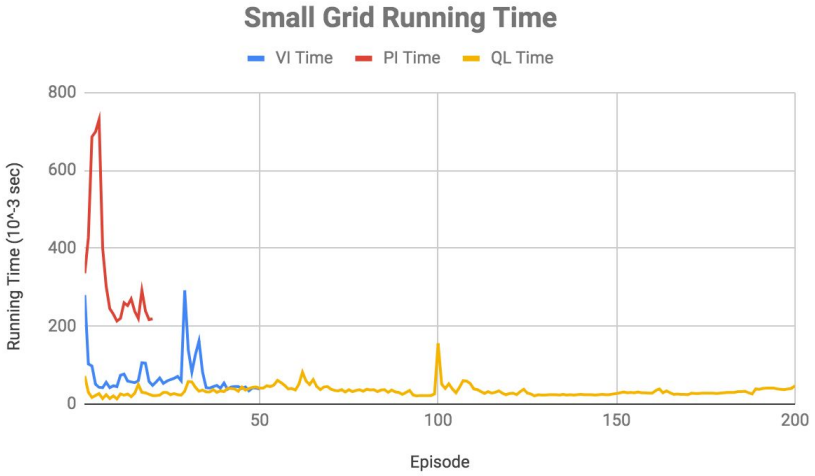


Q Learning was first run on the small grid for 500 episodes with learning rate 0.1. We plotted the number of steps in each episode in blue (left vertical axis) and running time in red (right vertical axis) as above. From the plot we have the following observations:

(1) About the first 50 episodes of Q Learning used up all 25 steps, meaning that none of them got out of the maze successfully. The number of steps goes smaller to 14 in episode 53 for the first time, but then quickly goes back to 25 again.

(2) The blue curve oscillate a lot between 25 and smaller number of steps. As the iteration goes on smaller numbers of steps show up more often, as we can see on the right side of the chart the blue curve stays between 10 and 15 more much more often than when between episodes 100-200. This shows that QL converges slowly on this MDP.

(3) In general, the running time stays about constant, with a few peaks. If comparing the the running time for VI and PI, we find that QL is quite fast as shown in the following figure.

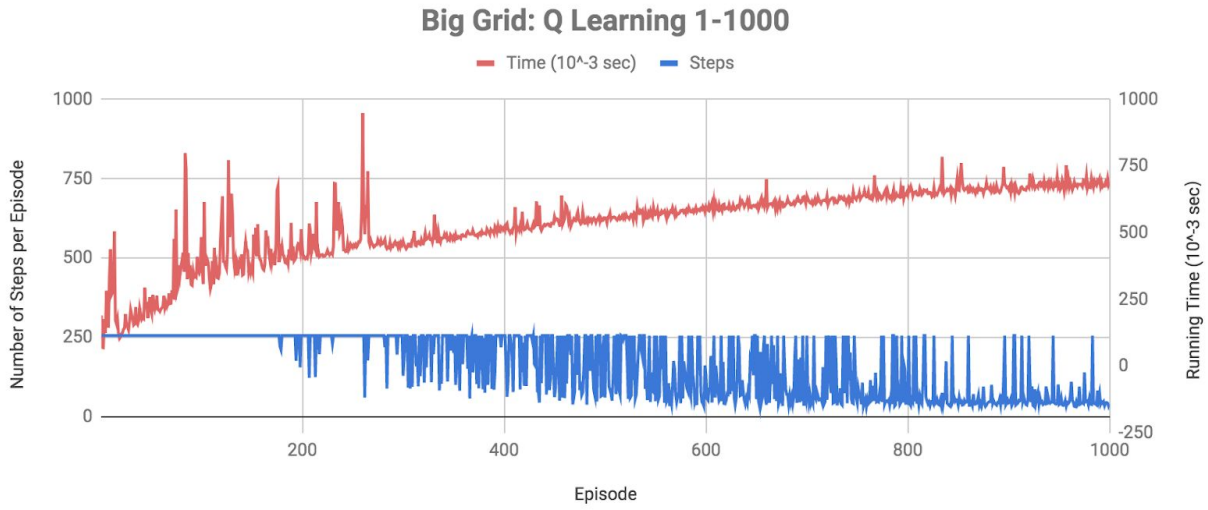


We now try to explain the above observations by comparing model-free RL algorithms and model-based RL algorithms. In the previously discussed model-based RL algorithms, the computation in each episode uses a lot of information of the MDPs, including the reward

functions, the distribution of different types of cells in the grid, what the neighbors of each cell are, and etc. However, in this QL algorithm which is model-free, we have no such domain knowledge as listed above for the model-based algorithms, and the only known at the beginning of the algorithm is the set of states of the MDP and the set of possible actions for each state. Due to the lack of domain knowledge, it is within our expectation that QL converges very slowly comparing to VI and PI, which explains (1) and (2).

Note that the lack of domain knowledge means fewer things to take care of during the computation, which explains the shorter computation time per episode in QL than VI and PI, as shown in the running time figure. And for the peaks shown in the QL running time curve in the first plot in this section, we observe that they actually match the drops in the number of steps (the blue curve), which is similar to what we observed for the PI running time curve. We believe this is again because being able to get out of the maze in a fewer number of steps makes some big change to the computation for the Q values for the next episode, which then increases the computation time. This explains (3).

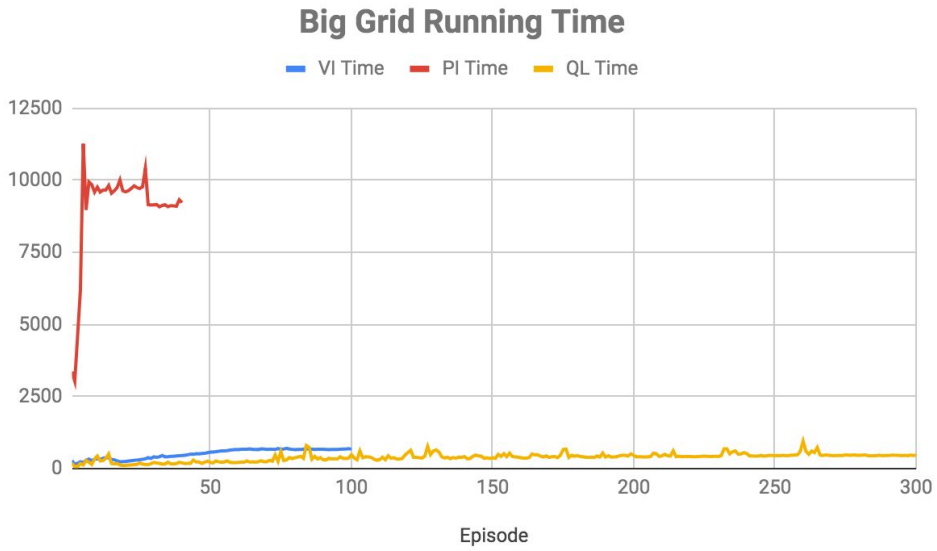
We then run QL on the big grid for 1000 episodes again with learning rate 0.1 and obtain plotted the results in a similar way as above.



Again, we observe that the algorithm starts to get out of the maze successfully a little before episode 200. The performance then starts to oscillate between 256 and smaller number of steps with the most of the episodes gets small number of steps out gradually. The running time plot looks a little different this time, as we see that the overall running time gradually goes up as the algorithm converges better and better between episode 400 and 10000, whereas in the running time plot for the small grid the time stays about constant till the end. We believe this is related to the core of the QL algorithm again. Recall when computing the Q value for a pair (s_t, a_t), we need to compute the maximum of $Q(s_{t+1}, a)$ over all choices of actions a, and then use the discount factor to multiply by this maximized value and proceed to the rest of the computation. The increase in the running time in each episode as the algorithm converges is probably due to that there is an

increasing number of distinct Q values for the (s, a) pairs as the algorithm goes on, leading to a bigger running time when computing the maximum of $Q(s_{t+1}, a)$ over all choices of actions a.

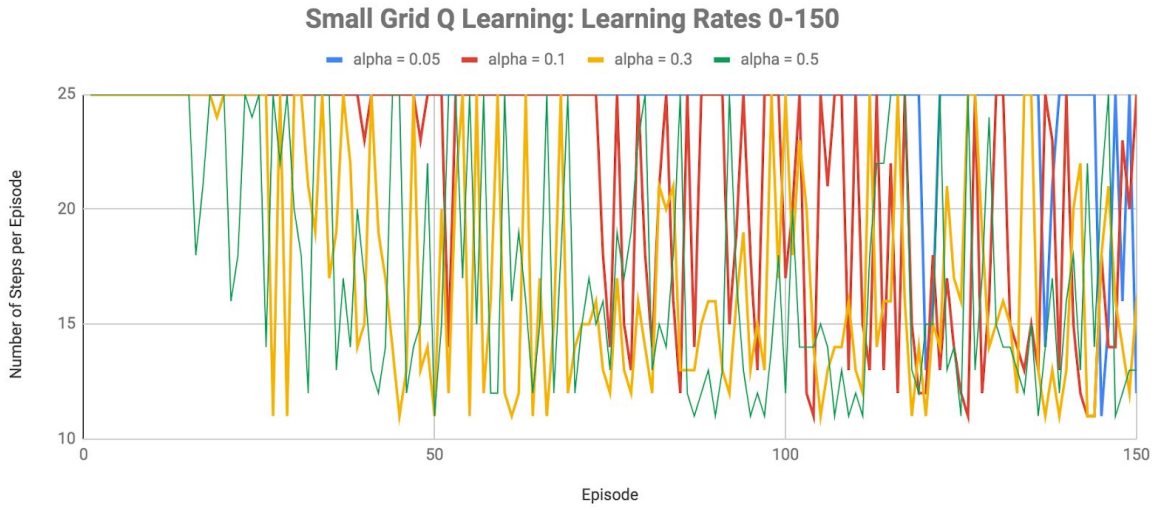
We also compare the running time per episode for all three algorithms on the big MDP and plot the result as follows.

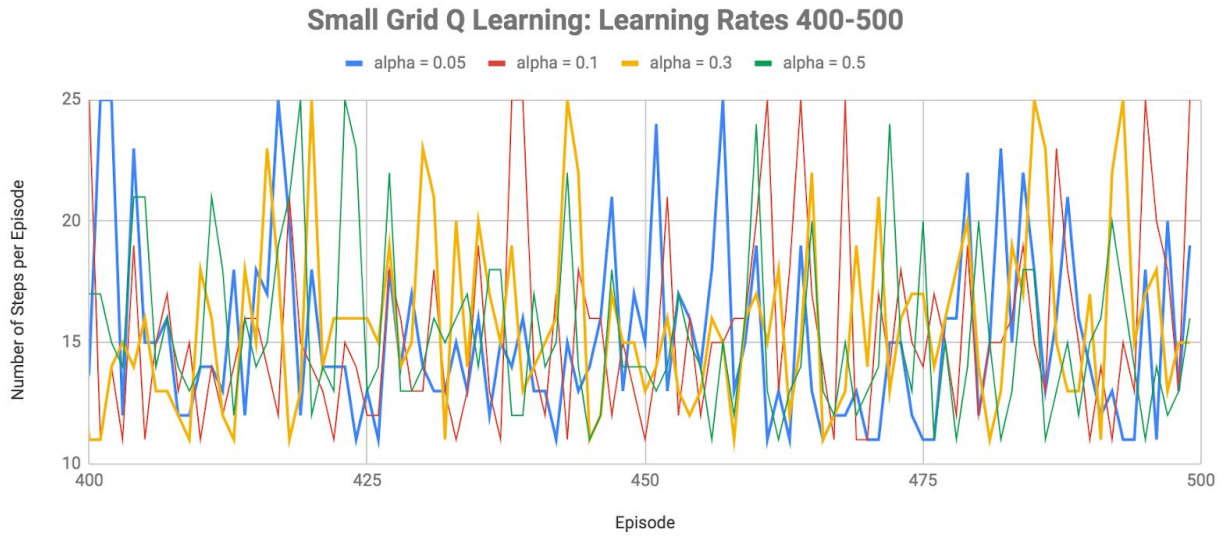


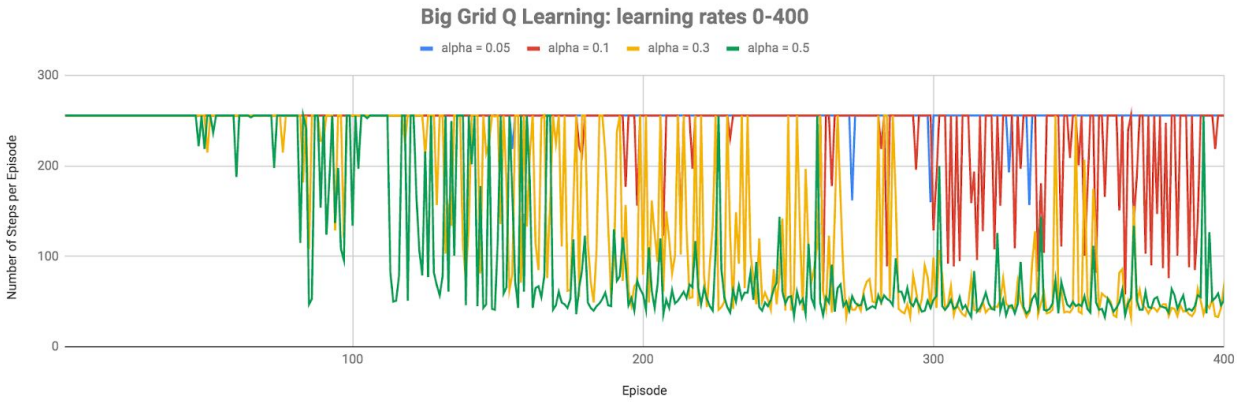
Again we observe that the running time per episode for VI and QL are similarly low whereas PI has a high running time in every episode.

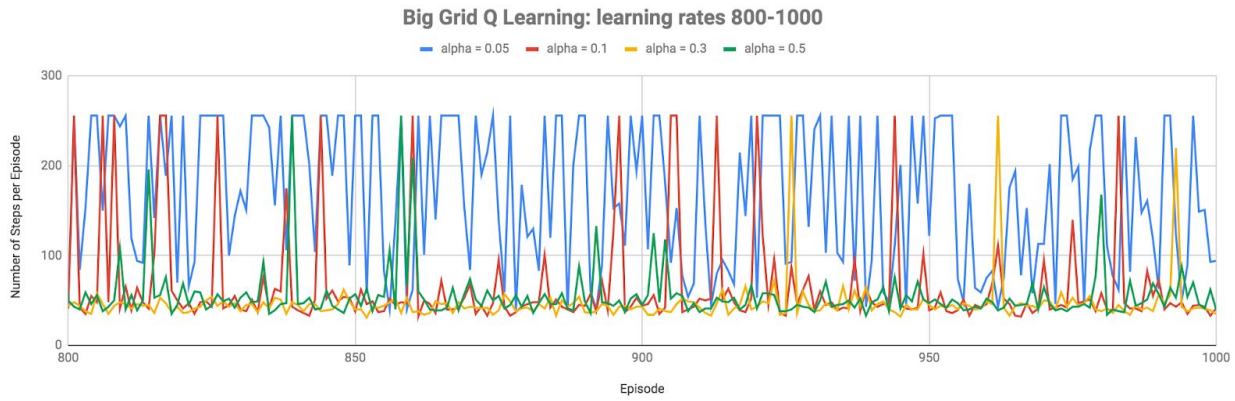
Q Learning with Various Learning Rates (alpha)

Note that the previous QL experiments on both MDPs used learning rate alpha = 0.1. In this section, we vary the learning rate for each MDP and observe the performance of the algorithms. In the following four plots, we experimented the performance of the QL algorithm with alpha = 0.05, 0.1, 0.3, and 0.5 for both MDPs. For each MDP, we get one plot at the beginning of the algorithm (episode 0-150 for small grid and 0-400 for big grid) and another plot by the end of the algorithm (episode 400-500 in small grid and 800-1000 in big grid).


Small Grid Q Learning: Learning Rates 0-150
alpha = 0.05
alpha = 0.1
alpha = 0.3
alpha = 0.5
Number of Steps per Episode
25
20
15
10
0
50
100
150
Episode


Small Grid Q Learning: Learning Rates 400-500
alpha = 0.05
alpha = 0.1
alpha = 0.3
alpha = 0.5
Number of Steps per Episode
25
20
15
10
400
425
450
475
500
Episode


Big Grid Q Learning: learning rates 0-400
alpha = 0.05
alpha = 0.1
alpha = 0.3
alpha = 0.5
Number of Steps per Episode
300
200
100
0
100
200
300
400
Episode

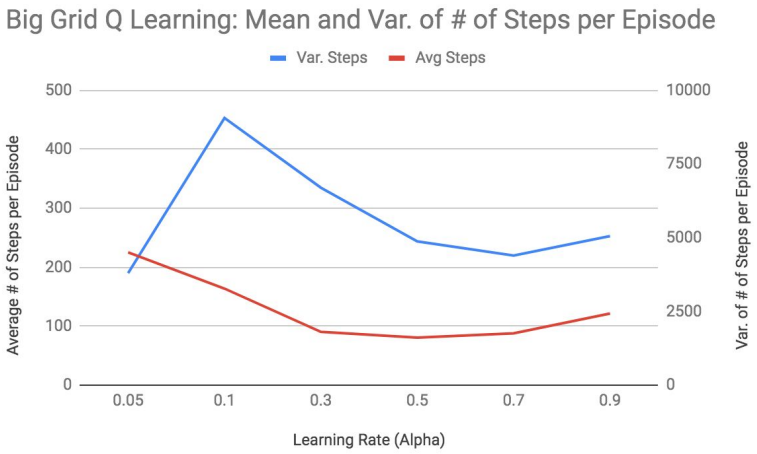


From the plots, we have the following observation and analysis.

(1) In both the first and the third plots, we observe that the green curve (alpha = 0.5) starts to converge to first, followed by the yellow curve (alpha = 0.3), the red curve (alpha = 0.1), and finally the blue curve (alpha = 0.05). Recall that the higher the alpha in Q Learning, the more the algorithm trusts the learned result from the previous episode. Thus it is within our expectation that the experiment with large learning rate starts converge early.

(2) In the second and the fourth plots above, we observe the performance of the algorithm at the late stage of the experiments. As we see in the fourth plot, the blue (alpha = 0.05) and red (alpha = 0.1) curves still oscillate a lot between 256 steps (the worst scenario) and smaller number of steps, whereas the yellow (alpha = 0.3) and the green (alpha = 0.5) curve show less up and down. This is because when alpha is no more than 0.5 for the large MDP, a higher value of learning rate leads to faster convergence. A similar observation can be made in the second plot too, except for that the green curve (alpha = 0.5) seems to have a bit more oscillation than the yellow curve (alpha = 0.3). This is probably because 0.5 as a learning rate for the small MDP is already too big, meaning that we should probably not trust the learned result in each iteration as much as 50%.

We further explored the mean and variance of performance per episode with more values of alpha. The results are plotted as follows.



The means of the number of steps per episode are plotted in red (left vertical axis) and the variances are plotted in blue (right vertical axis). We observe that in the small MDP, the minimum mean is achieved when the learning rate is between 0.3 and 0.5, with the low of variance achieved at learning rate 0.3. Note the variance for alpha = 0.3 being lower the the variance alpha = 0.5 is consistent with our observation (2) above. In the large MDP, the low of the mean is achieved around 0.5 and the low of the variance is achieved between 0.5 and 0.7. Too big a learning rate would result in too much trust if the learned result in each episode, which is similar to ``overfitting'' that we discussed in the section of supervised learning. We conclude the small grid and the big grid have the best performance when the learning rate is around 0.3 and 0.5, respectively.