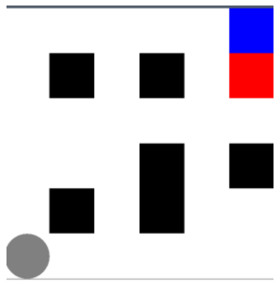
For this assignment, I will use the grid-world problems (1 easy and 1 hard) in order to compare different Reinforcement Learning algorithms : Value Iteration, Policy Iteration and Q-Learning. The problem generation and the 3 algorithms will be implemented based on Jonathan Tay's code, Yan Cai's code and BURLAP project (link at the end of the document) using Jython. The easy grid-world problem is a 6x6 grid with 1 hole and the hard grid-world problem is a 15x15 grid with 4 holes.

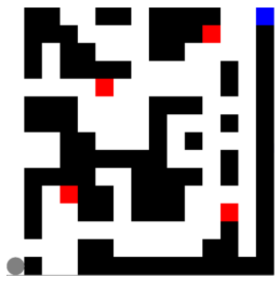
The grid-world problem is similar to the class' example where we have to find a way to travel from a starting state to a destination state. We can only move in 4 directions : up, down, left and right to move from 1 state to another state. The grid-world will have 'walls' which will restrict our possible travel directions and also have 'holes' which we need to avoid while travelling to the destination point.

In order to go from our starting state to the destination state, we are using rewards to find the best way for our travel plan. Our destination state has a reward value 100. All other states except the destination and the 'holes' will have a reward value of -1 and the 'holes' state has a reward value of -50. The goal of the algorithm is to find the travel plan that will result in the highest total reward.

Our starting point (grey circle) will be the bottom left state and our destination point will be the upper right state (blue square). 'Walls' is visualized as black square and 'holes' is visualized with red square. The probability to move to the intended direction is 0.7 and the probability it will move to the each other 3 directions is 0.1, so when we decide to go up, there is a 0.3 probability that it will move to the other 3 directions.

The 2 MDP grid-world problems that I will use :


MDP easy grid problem


MDP hard grid problem

For my experiment, I will run the analysis of the 3 algorithms for the easy grid-world problem and then use the same algorithms for the hard grid-world problem. For the Q-learning algorithm, I will also vary the learning rate (0.1, 0.5 and 0.9) and the epsilon (0.2, 0.8) to see if it will affect the convergence of the Q-learning algorithm. I also set a max iteration of 100 for Value and Policy Iteration, and max iteration of 1000 for the Q-learning. I set more iterations for the Q-learning algorithm because the Q-learning algorithm might need to do more iterations before convergence because of the stochastic property it has for doing random-restart and having a certain learning rate value.

For the 3 algorithms, I will consider convergence when the number of steps taken and the total reward value does not change significantly for future iterations. For Value iterations, this will mean that the states value does not change significantly over iterations. For Policy iterations, this will mean that the policy used does not changed over iterations and thus the total utility value does not change significantly. As for Q-learning, since it will be hard to determine the real convergence (when all possible state-action pairs are visited long enough), thus determining convergence when the number of steps taken and the total reward value does not change seems to be reasonable.

First of all, I will analyze the easy grid-world problem.

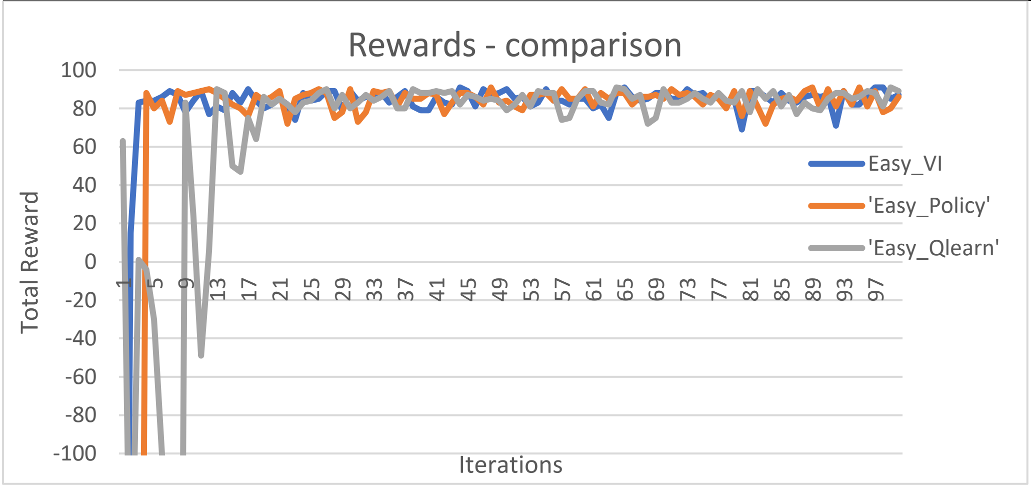
The comparison below, I will use Q-learning algorithm with learning rate = 0.1 and epsilon 0.8 since it seems to have the highest average total reward and lowest average total steps out of the 6 possibilities.

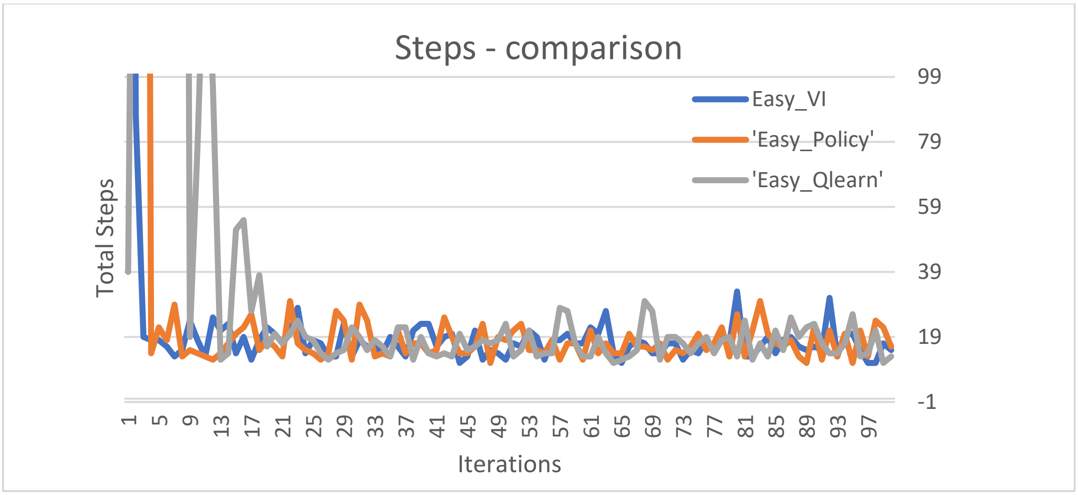
Q-learning average results from 1000 iterations – highlighted means best :

average	time	reward	steps	delta convergence
Q lean lr 0.1 epsilon 0.2	0.22398483	77.821	22.023	1.951197053
Q lean lr 0.1 epsilon 0.8	0.481511862	79.835	19.911	2.162996707
Q lean lr 0.5 epsilon 0.2	0.255425778	55.071	43.842	8.630966498
Q lean lr 0.5 epsilon 0.8	0.255425778	55.071	43.842	8.630966498
Q lean lr 0.8 epsilon 0.2	0.291699714	30.482	62.943	21.23022898
Q lean lr 0.8 epsilon 0.8	0.735165683	4.4	80.303	24.24444628

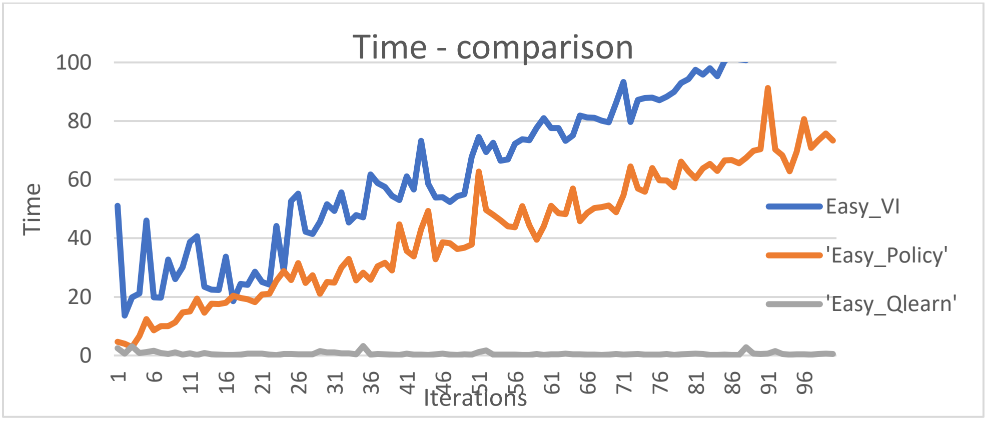
Comparison between the 3 algorithms for rewards and steps convergence for 100 iterations:

	Value Iteration	Policy Iteration	Q-learn
Iter until reward convergence	8/9	8/9	20
Iter until steps convergence	8/9	8/9	21/22

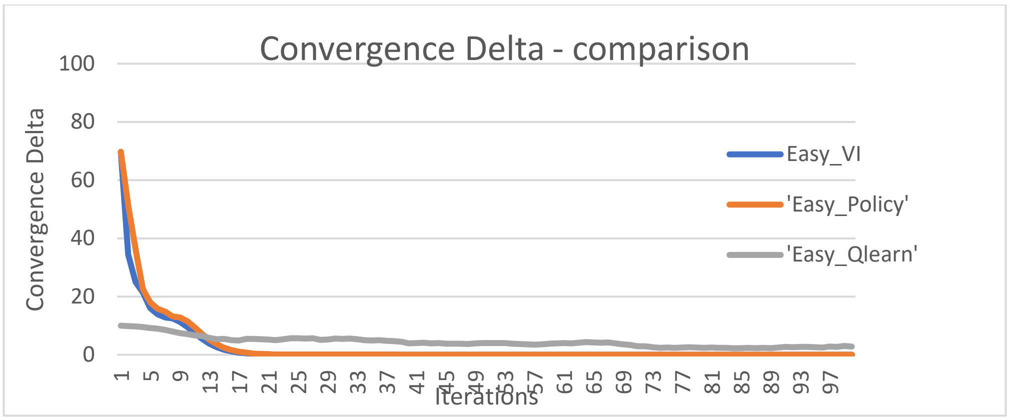




From the 2 charts above, it seems that Value iteration and Policy Iteration converges around the same iterations (around 8/9 iterations) for both the steps and total rewards while the Q-learning algorithm with learning rate = 0.1 and epsilon = 0.8 converges after around 20-22 iterations. This is to be expected since Q-learning algorithm does not have information about the reward function and the states transition of each state. Also, Q-learning algorithm do some random actions to find the optimal policy which require larger amount of iterations in order to converge.



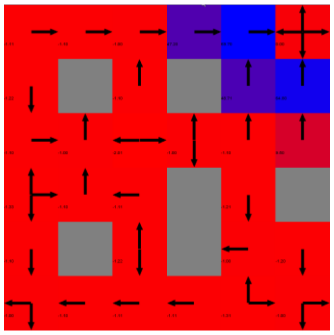
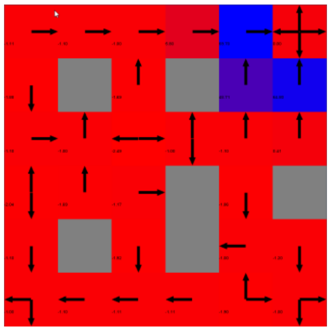
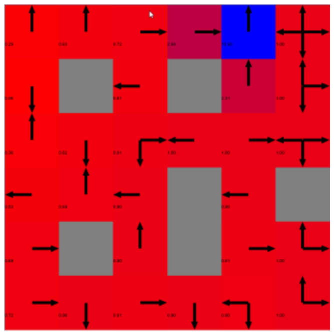
From the time-comparison chart above, it seems that Value and Policy iteration algorithm takes more time compared to the Q-learning algorithm. This is as expected because both Value and Policy iteration needs to re-evaluate and re-calculate the utilities and policy for each iteration, while Q-learning does not to do the same amount of re-evaluation and re-calculation for each iteration because of the 'exploitation' of previous iterations with probability 0.8 (epsilon-greedy).



Lastly, the convergence delta – comparison chart will show the difference between the algorithm value function using Bellman equation and our current policy. From the chart above, it seems like both Value and Policy iteration converge to a delta value close to 0 around the same iterations (around 12/13 iterations) for the convergence delta. While the Q-learning algorithm starts really well, the convergence delta improvement is much slower compared to the other 2 algorithms. This could be caused by the exploitation vs exploration properties that it has that caused the algorithm to not explore far enough.

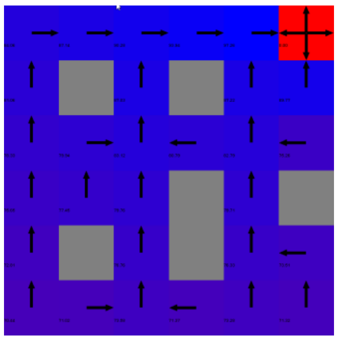
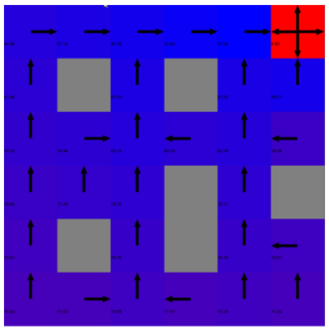
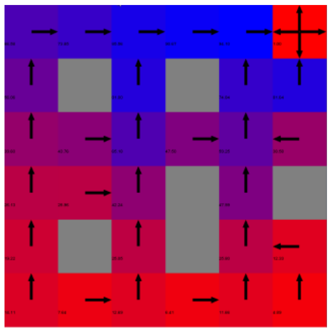
To compare the final answer for the 3 algorithms, below are the comparison of the final policy at iteration 1, 50 and 100 :

Iteration 1 :

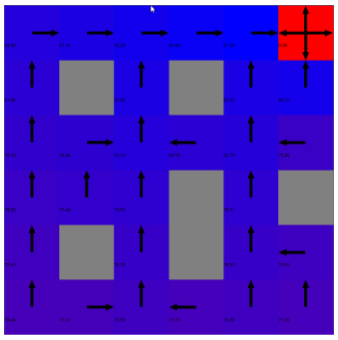
Value Iteration Policy Iteration Q-learn lr=0.1, eps=0.8

Iteration 50 :

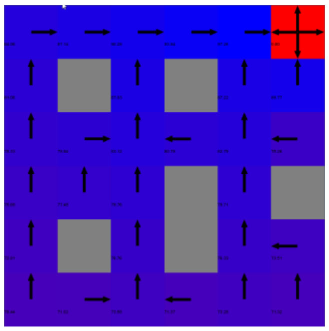




Value Iteration Policy Iteration Q-learn lr=0.1, eps=0.8

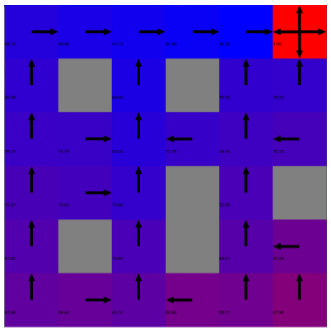
Iteration 100 :



Value Iteration

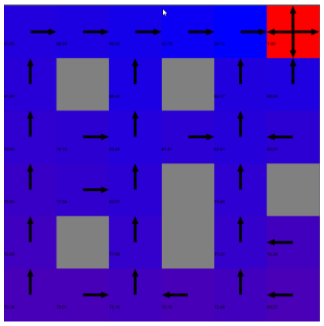


Policy Iteration



Q-learn lr=0.1, eps=0.8

Iteration 1000 for Q-learning :



From the screenshots above, it seems like the Value and Policy algorithm converges to the same answer while the Q-learning algorithm provides a slightly different answer (1/2 state have different actions).

Next, I will analyze the hard grid-world problem. Using the same steps as the easy problem.

The comparison below, I will use Q-learning algorithm with learning rate = 0.1 and epsilon 0.2 since it seems to have the highest max total reward and lowest average total steps out of the 6 possibilities. I used max rewards and min time, steps and delta convergence instead of average because the data has a lot of spikes that could be caused by the hard-problem domain.

Q-learning average results from 1000 iterations – highlighted means best :

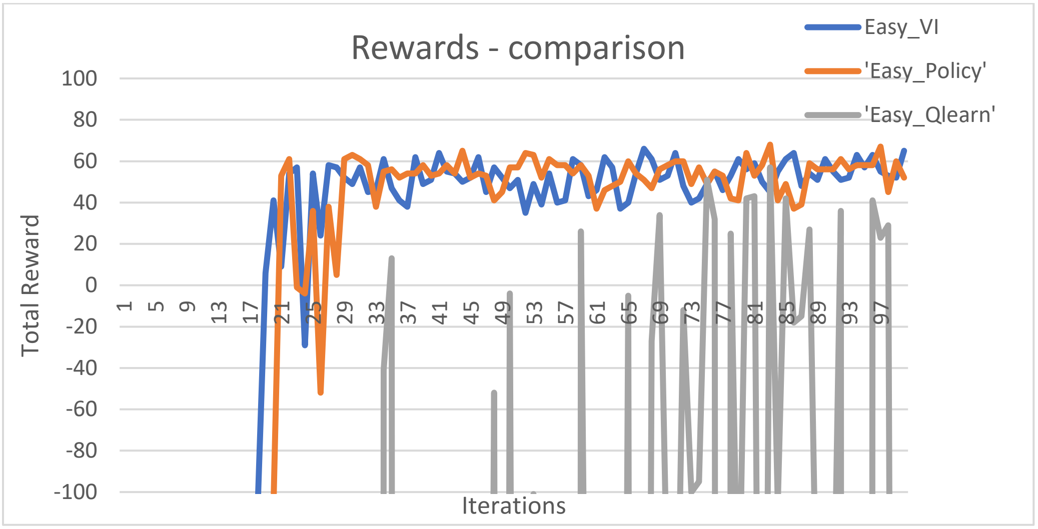
Max and Min	time	reward	steps	delta convergence
Q lean lr 0.1 epsilon 0.2	0.212604	71	31	0.656036099
Q lean lr 0.1 epsilon 0.8	0.491578	70	31	3.911439354
Q lean lr 0.5 epsilon 0.2	0.1951	70	32	6.503532796
Q lean lr 0.5 epsilon 0.8	0.751589	68	34	11.5005
Q lean lr 0.8 epsilon 0.2	0.256729	64	38	17.99181
Q lean lr 0.8 epsilon 0.8	0.723144	59	43	17.99181

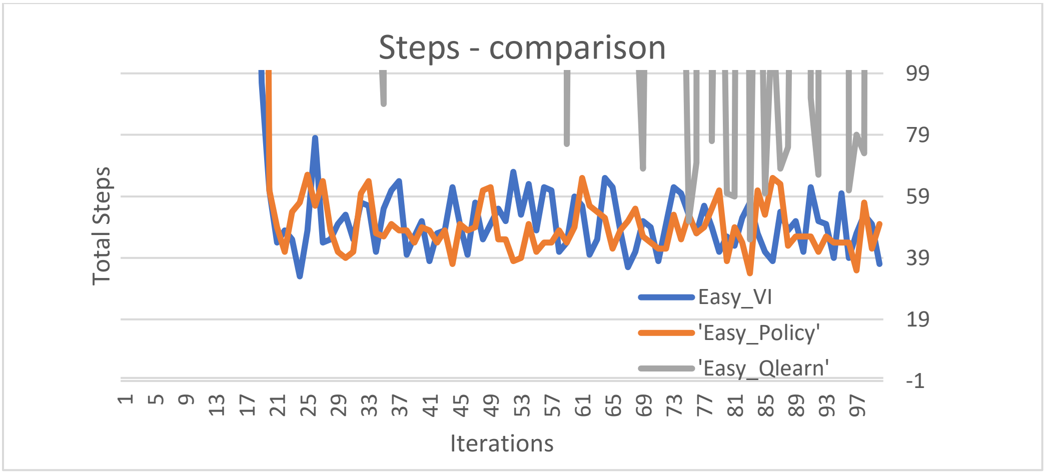
Comparison between the 3 algorithms for rewards and steps convergence for 100 iterations:

	Value Iteration	Policy Iteration	Q-learn
Iter until reward convergence	28/29	29/30	N/A (> 100)

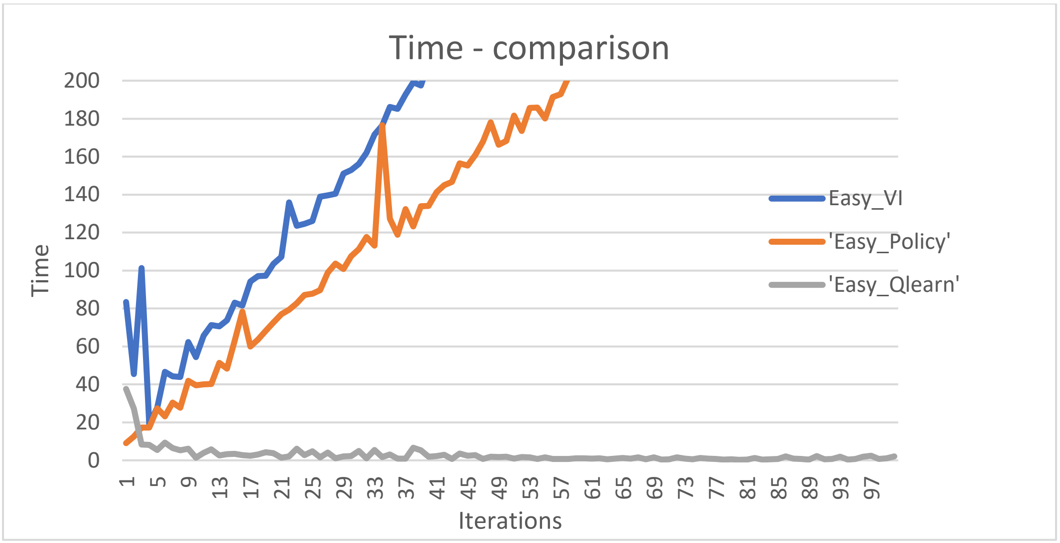
Iter until steps convergence	29/30	29/30	N/A(> 100)

For the Q-learning algorithm, it does not seems to converge after 100 iterations.

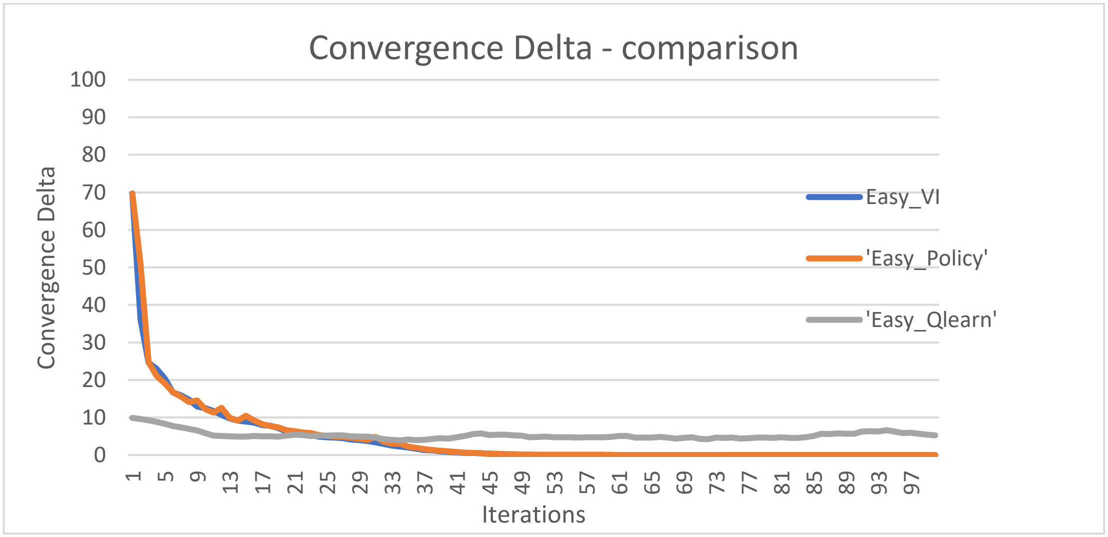




From the 2 charts above, it seems that Value iteration and Policy Iteration converges around the same iterations (around 28-30 iterations) for both the steps and total rewards while the Q-learning algorithm with learning rate = 0.1 and epsilon = 0.2 does not really converge after 100 iterations. This could be caused by the hard-problem domain that requires Q-learning algorithm to do a lot more iterations because now the grid world is 15x15 instead of 6x6 and have more 'holes' which and thus the current grid world has a lot more possibilities to cover for the Q-learning algorithm because it does not know the reward function and the states transition of each state.



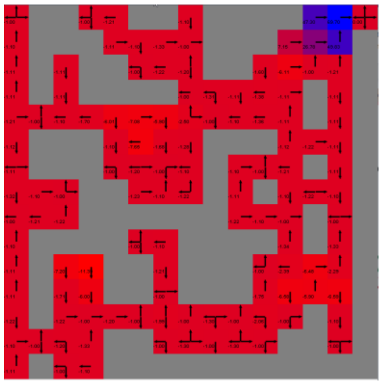
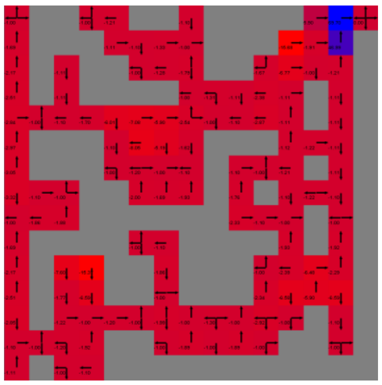
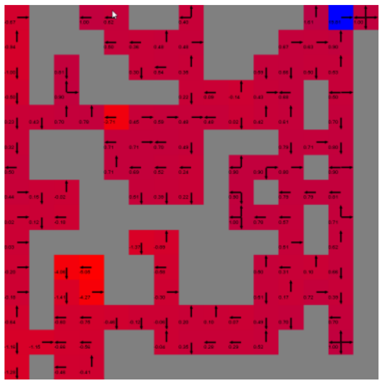
From the time-comparison chart above, it seems that Value and Policy iteration algorithm takes a lot more time compared to the Q-learning algorithm. This is as expected because both Value and Policy iteration needs to re-evaluate and re-calculate the utilities and policy for each iteration and with the number of states increased significantly (6x6 vs 15x15), re-calculation and re-evaluation will take much more time. Same with the easy problem explanation, Q-learning does not to do the same amount of re-evaluation and re-calculation for each iteration because of the 'exploitation' of previous iterations with probability 0.2 (epsilon-greedy)



Lastly, the convergence delta – comparison chart will show the difference between the algorithm value function using Bellman equation and our current policy. From the chart above, it seems like both Value and Policy iteration converge to a delta value close to 0 around the same iterations (around 40-42 iterations) for the convergence delta. While the Q-learning algorithm starts really well, the convergence delta improvement is much slower compared to the other 2 algorithms and is much slower compared to the easy problem. This is as expected because with the hard problem, there are a lot more possible states and thus Q-learning algorithm will need much more iterations in order to converge

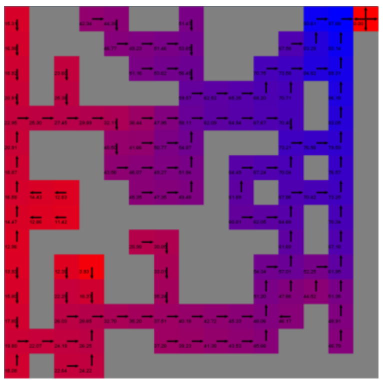
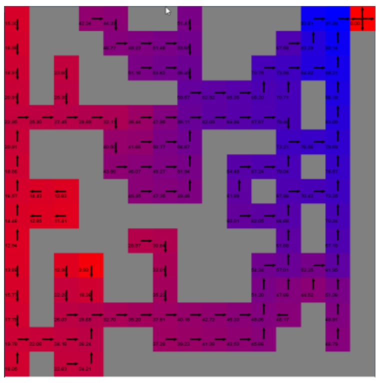
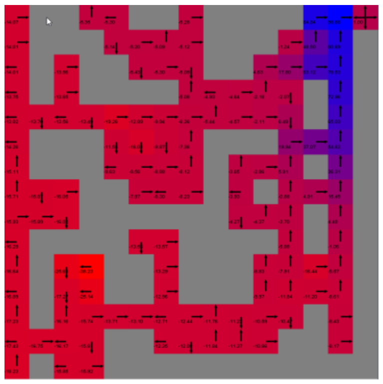
To compare the final answer for the 3 algorithms, below are the comparison of the final policy at iteration 1, 50 and 100 :

Iteration 1 :

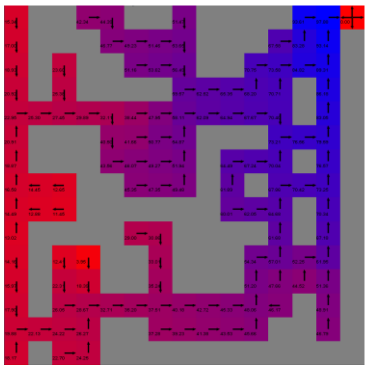
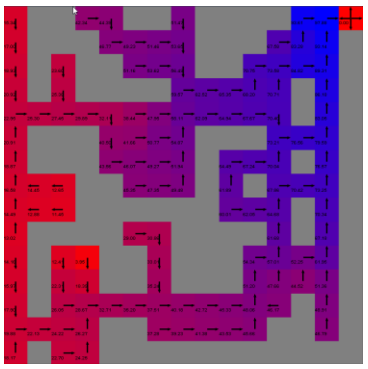
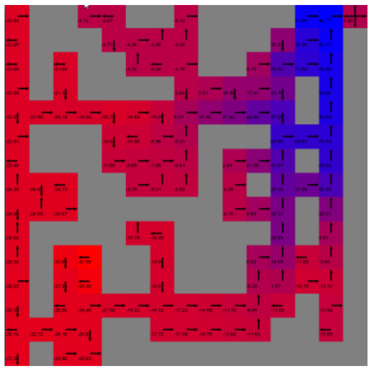
Value Iteration Policy Iteration Q-learn lr=0.1, eps=0.8

Iteration 50 :

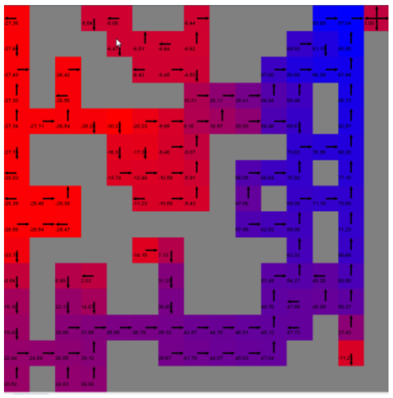
Value Iteration Policy Iteration Q-learn lr=0.1, eps=0.8

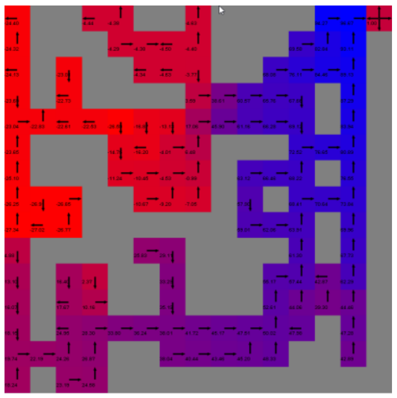
Iteration 100 :

Value Iteration Policy Iteration Q-learn lr=0.1, eps=0.8

Iteration 1000 for Q-learning : Iteration 10000 for Q-learning :





From the screenshots above, it seems like the Value and Policy algorithm converges to the same answer while the Q-learning algorithm does not really converge to the correct answer even after 10000 iteration. The Q-learning algorithm seems to be confused for the upper part of the problem where there are dead-ends. This could be solved by increasing the number of maximum iterations or by exploring more learning rate and epsilon values.

From the 2 experiments above, it seems that both Value and Policy iteration algorithms are guaranteed to converge around the same time and produces the same final solution for both easy and hard problem while the Q-learning algorithm converged for the easy problem with more iterations compared to the Value and Policy Iterations but failed to converged for the hard problem after 100,1000, and 10000 iterations due to the increase of states and complexity. Although the Value and Policy iteration converged for both problems, the time it takes per iteration increase significantly compare to the Q-learning algorithm which could cause a problem when the problem gets bigger and much more complicated.

Lastly, from the experiments above, we can see that as we increase the number of states and complexity of the problem (by adding more ‘holes’), all 3 algorithms performed worse, especially for Q-learning algorithm since it now has to do much more iterations to converge.

Coding reference :

https://github.com/JonathanTay/CS-7641-assignment-4

https://github.com/danielcy715/CS7641-Machine-Learning

https://github.com/jmacglashan/burlap

http://burlap.cs.brown.edu/