Assignment 4: Markov Decision Processes

Minming Zhao (gtID: 902685774)

Problem statement

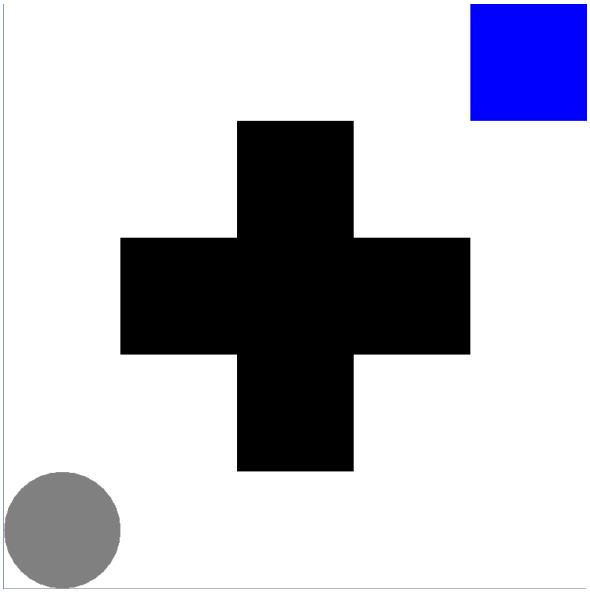
In this assignment, two MDP (Markov Decision Processes) are discussed and called Grid World. One MDP has a small number of states and the other has a large number of states. This problem contains an agent (gray circle), walls (black squares), and a target (blue square) which is a terminal state. I tried to make these two MDP as similar as possible to study the effect of having different number of states. The objective of this MDP is to move the agent from initial state which is at the bottom left corner to the terminal state with minimal steps as possible.

To illustrate the visualization of this problem, the visual representation of the two MDP problems has been showed below. At the left, it is the easy version MDP and it is 5X5 gird and contains 20 states including initial and terminal states. At the right, it is the complex version MDP grid, which has 11X11 grids and contains 96 possible states including initial and terminal states.

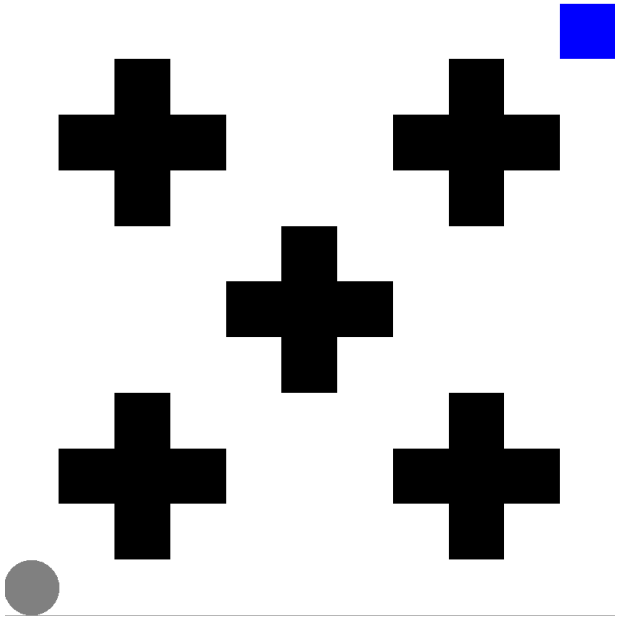
The movement of agent can be north, east, south and west (or up, right, down, left from paper). The transition function is defined as 80% to the targeted direction, while 6.67% probability toward the other three directions. The reward function is defined as -1 for each step. Once the agent reaches the terminal state, it will be rewarded 100.

The code is java based and BURLAP (Brown-UMBC Reinforcement Learning and Planning) java code library is used.

Easy version MDP



Complex version MDP



MDP definition

States	**20 (easy version)** **96 (complex version)**
Model (Transition function)	$Pr(x\|x) = 0.8; Pr(x\|y) = 0.067.$ $x, y \in \forall$ distinct actions
Actions	**North, East, South, West**
Reward	**-1 for each step except terminal step rewards 100**

Why Grid world is interesting

Grid world is a very simple Markov Decision Process and is well explained in lecture. It may be far from any realistic situation. However, one might notice the circle, or the agent moves to avoid obstacles and tries to reach terminal state as small steps as possible, by which means that it tries to reach destination as fast as it can. The analogy could be applied for a self driving car problem, in a very simplified manner, trying to deliver passengers as fast as it can or as running shortest distance as it can, meanwhile, to avoid any obstacles as it can, thus do the job safely.

Hence, the grid world problem could be an over-simplified version of self-driving car problem in some sense. The discussion here in principle should be a fundamental piece of self-driving car in real world.

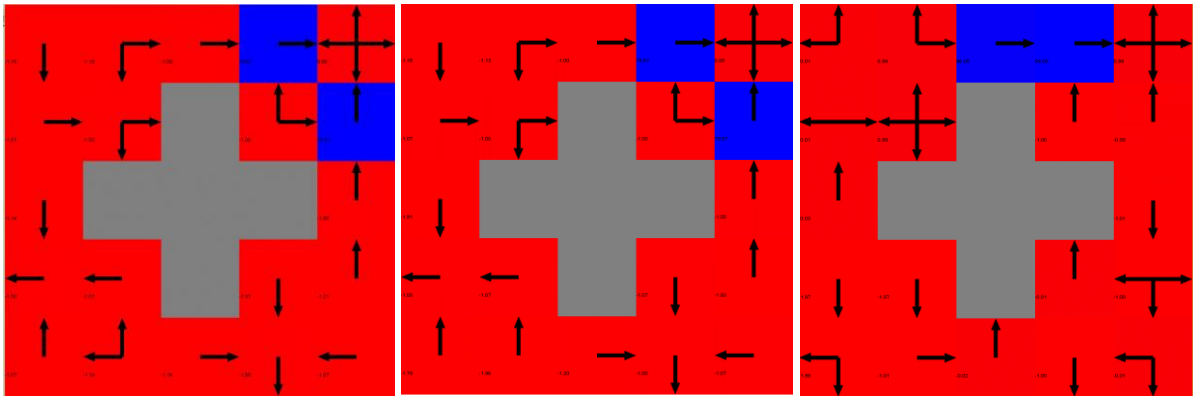
Grid World – easy version

To compare and evaluate the differences between value iteration, policy iteration and Q learning, an easy version of this MDP is discussed first. The first three images below represent the policy generated by the three alorithms after running after 1 iteration. As expected, Value Iteration and Policy Iteration have both generated very similar policies (actual identical in this case). Some states near the terminal states are surprising good for Value Iteration and Policy Iteration even with only 1 iteration while Q learning seems to suffer a bit as it seems more states have dead directed actions to the walls, obstacles or toward leaving the terminal states. The small number in each grid represents the utility of each states. Of course after 1 iteration, the utility has not been calculated well and does not mean too much to drive a real policy.

CS7641 Machine Learning

1 iteration:

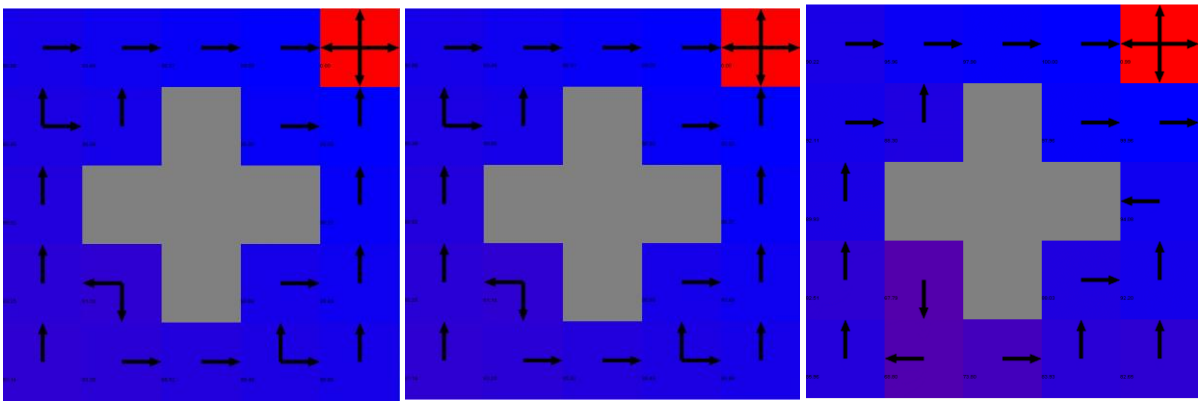
Value Iteration Policy Iteration Q Learning



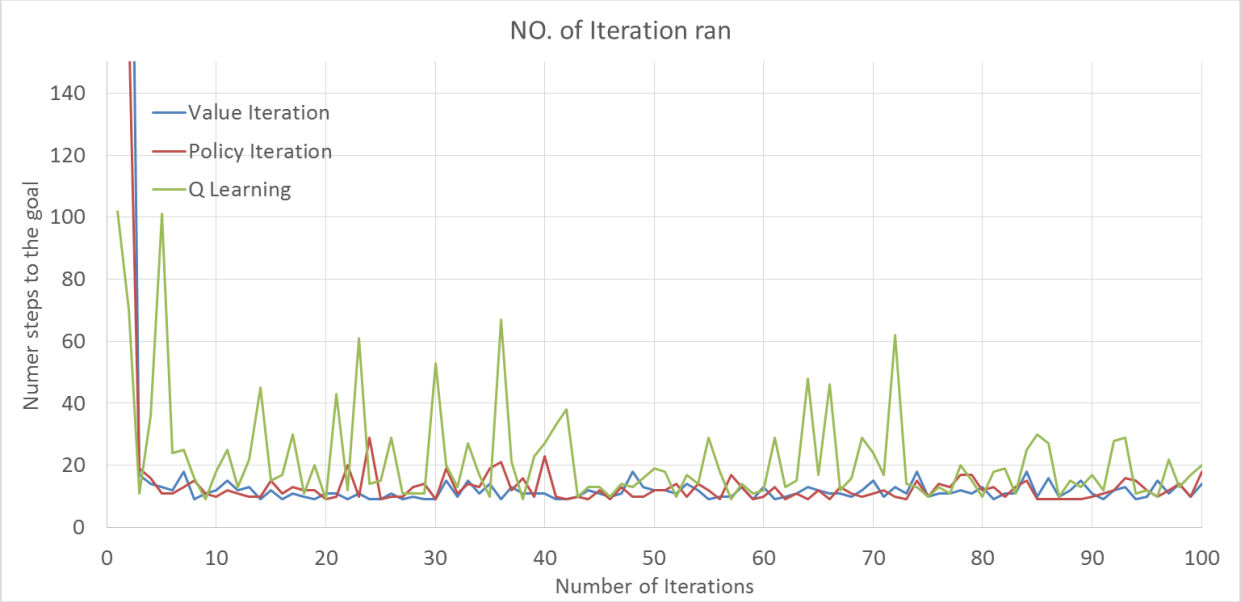
In this problem, I ran 100 iterations and we could see the policy change dramatically after 100 iterations. Value Iteration and Policy Iteration converged to the optimal policy and they are identical. One can notice the utility and policy are symmetric as the problem is symmetric. However Q Learning is less optimal, because the policy only leads to upper and right direction while neglect the symmetric right and upper way. Also the state on the south/underneath of terminal step shows arrows east/right which apparently leaves terminal state instead of directly leads into the final terminal state. Value iteration and policy iteration is at optimal policy while Q Learning is also optimal but less optimal.

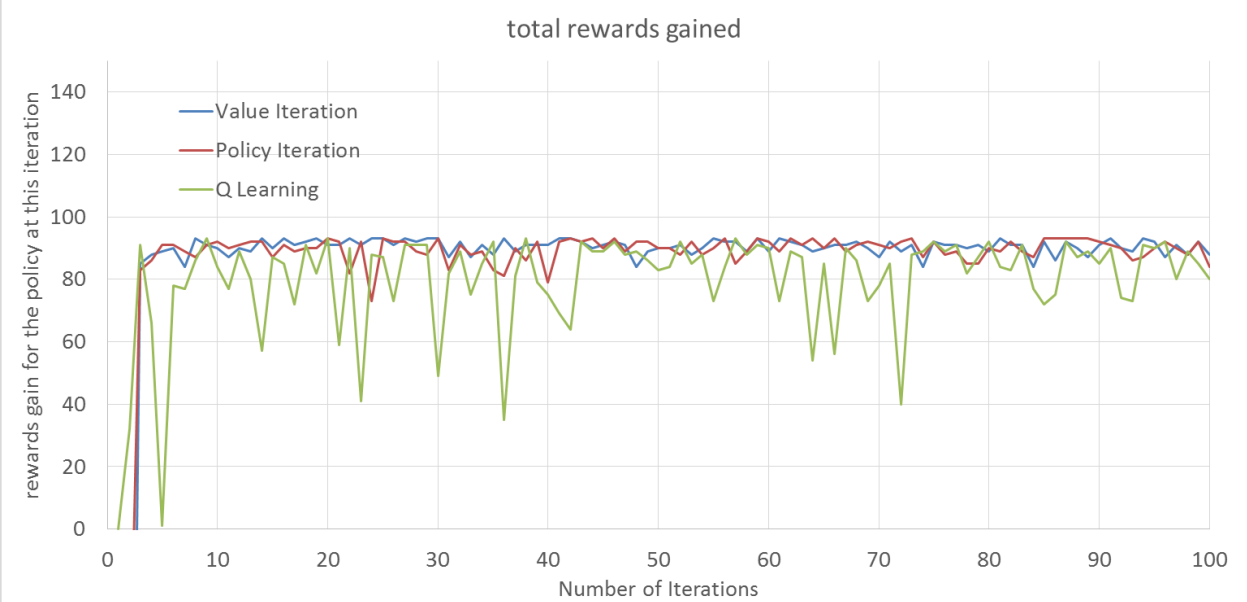
100 iterations:

Value Iteration Policy Iteration Q Learning

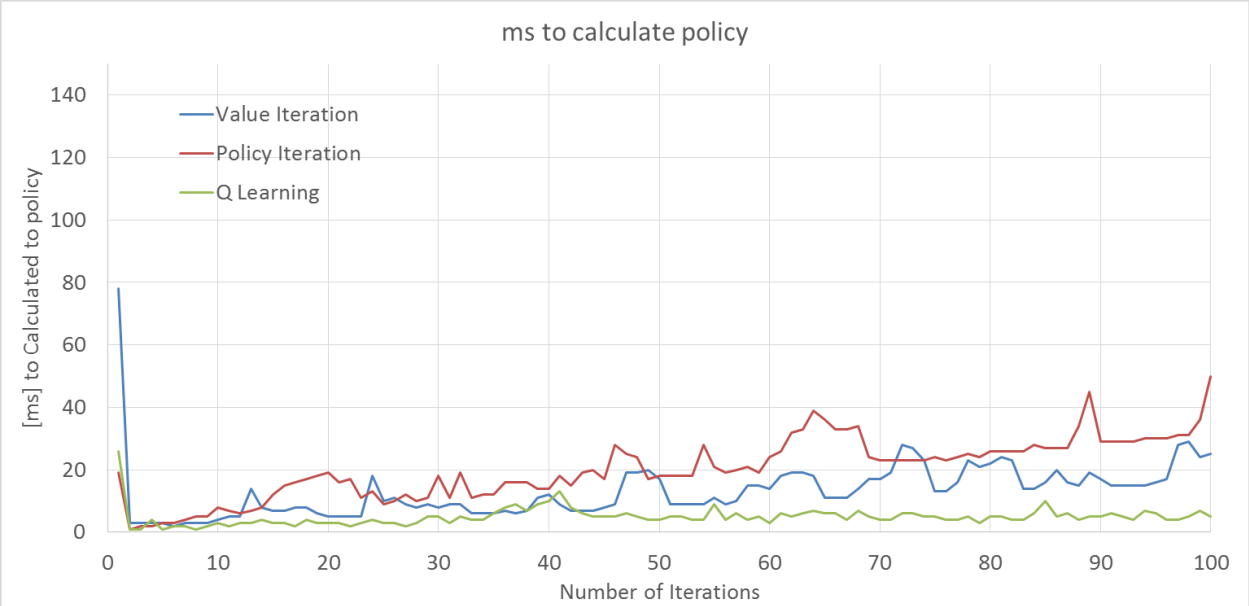


In order to study at which iteration each algorithm reach its convergence, the graph below is represented. Both plots tell similar information, Upper is to show the number of steps required to reach the terminal step after a certain number of iterations and the bottom one is to show the reward gain so far at each iteration for the optimal policy. We can notice that for Value Iteration and Policy Iteration, they are able to reach convergence at optimal policy around 4-7 iterations while Q Learning takes around 9 iteration. Also in terms of variability, Q Learning varies a lot even it once reached good convergence. It can be guessed that for the first tens of iteration, Q Learning is still exploring all states and it reach good convergence perhaps due to luck.





Also the processing speed of each algorithms are also studied and been plotted below. One can notice that Value Iteration and Policy iteration processing time increases linearly clearly as iteration grows. This can be understood as iteration grows, more states need to be recomputed accordingly. Among these two, it seems policy iteration takes more time than value iteration, it might be due to the fact that policy iteration require calculation from policy to utility and back calculate again from current utility to generate next policy, while value iteration only focus on utility until convergence. In contrast, Q Learning seems to hold a constant processing time around 5 millisecond per iteration. It might be due to the fact that Q Learning is not actually computing yet but rather simply trying the actions and rewards until a policy is calculated at the very end.



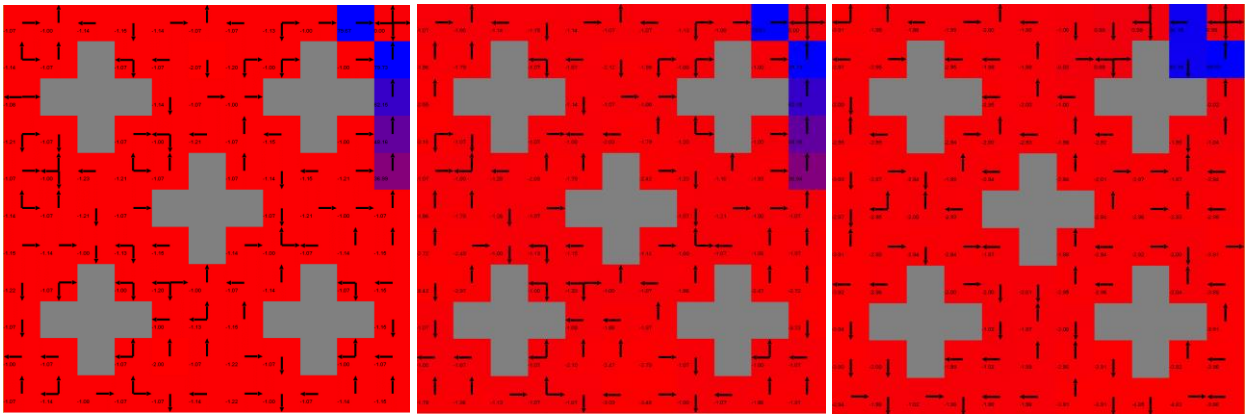
Grid World - complex version

In order to better differentiate the three algorithms in MDP problems, we increase the grid from initial 5X5 to 11X11, and the possible states increases from initial 20 to 96, the obstacle increases from initial 1 to 5 in total.

After 1 iteration, both Value Iteration and Policy Iteration seems to guess well 4-5 steps near the terminal state. Q Learning is more like random guessing and it could only guess well 1-2 steps away.

1 iteration:

Value Iteration | Policy Iteration | Q Learning

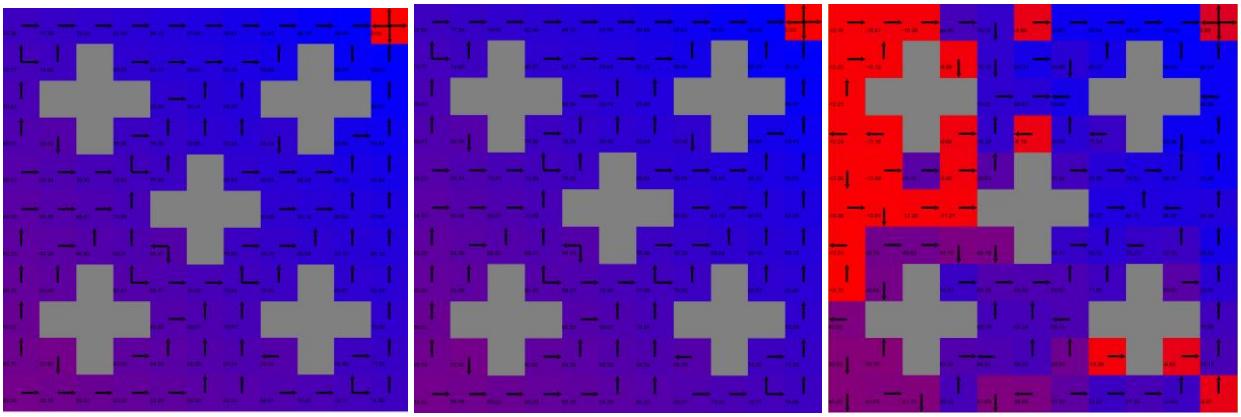


Next we run 100 iteration on this complex MDP to generate a new policy. The policy has been shown below. The Value iteration and Policy Iteration again gives almost identical policy and we could tell it should be the optimal one. However as we could see from easy version, the optimal policy was to go

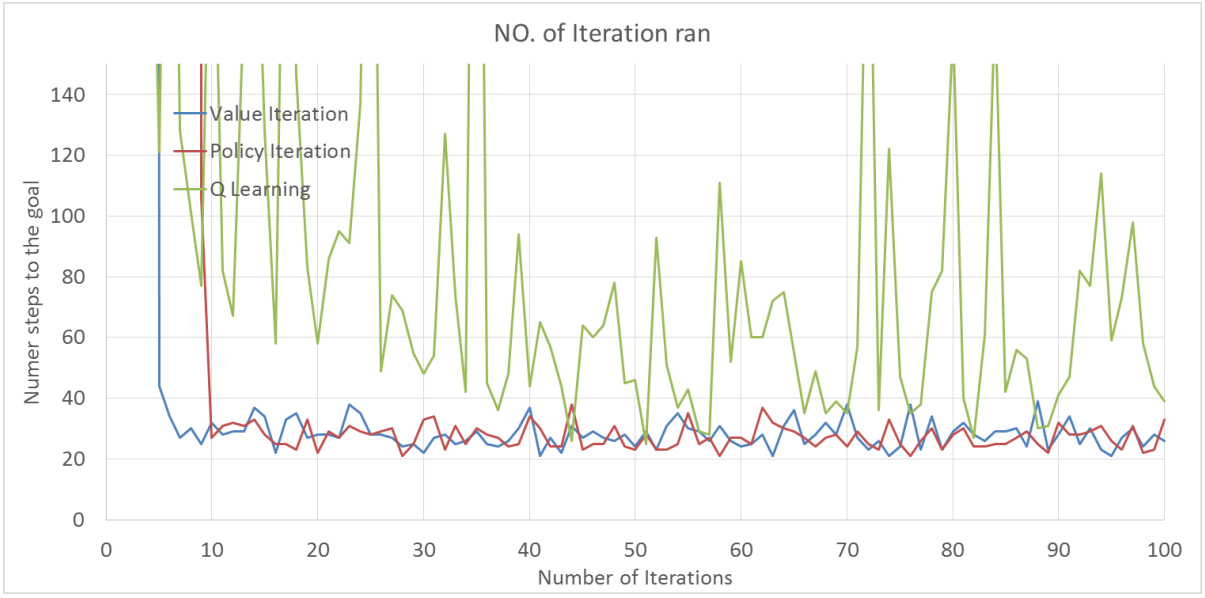
along the side of the big square while this optimal policy for complex one becom a bit zig-zag shape although it is the same reward. However in terms of Q Learning, this time Q Learning was not able to converge and more squre are in red which means they hold negative utility, i.e. no to enter those state at all, or actually there is no effective computation has been calculated at all for those states. This is worse than previous easy version Q Learning where it at least give a policy could leads to terminal state with less optimal but acceptable policy.

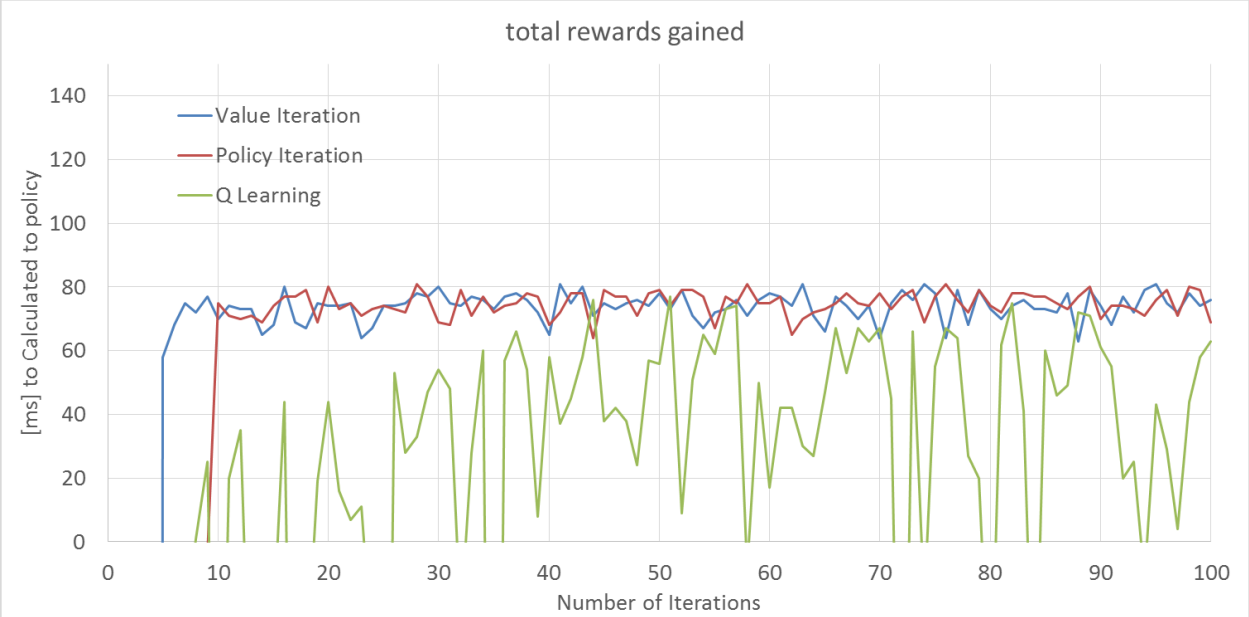
100 iterations:

Value Iteration | Policy Iteration | Q Learning

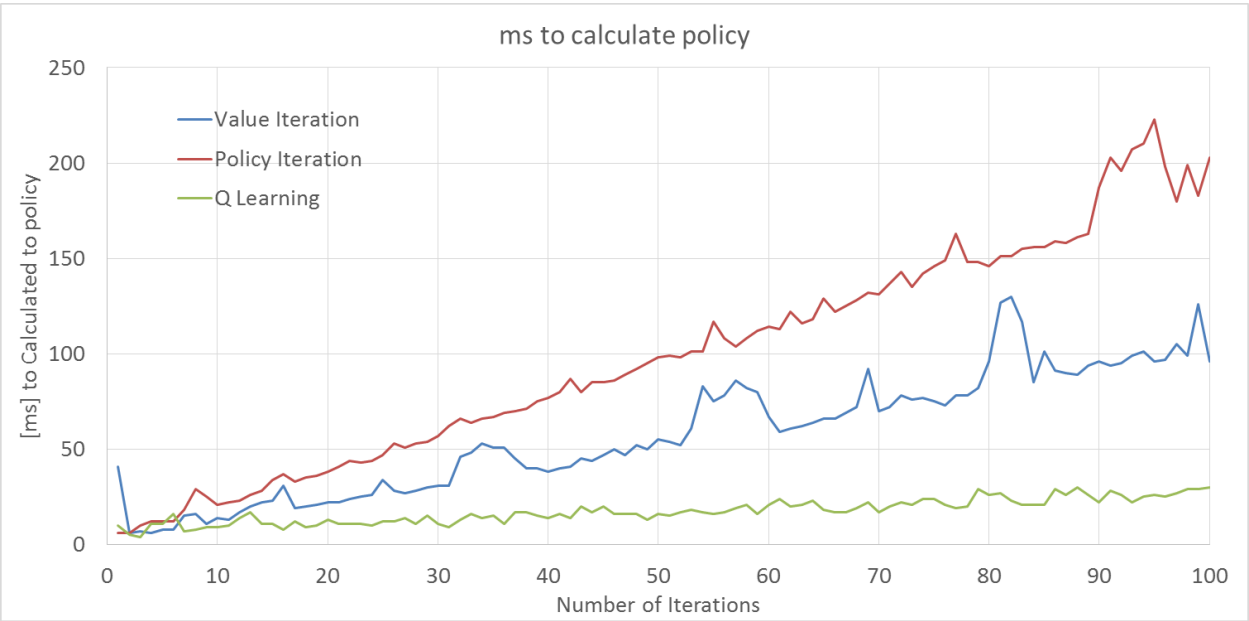


Similar to the previous study, the number of iterations needed to convergence and the total rewards gained per iteration are plotted below. As we can see, Value Iteration was able to achieve convergence at optimal policy at 7th iteration with 27 steps, while Policy Iteration achieve same optimal policy at 10th iteration with 27 steps. In contrast, Q Learning reach 26steps policy at 44th iteration and it suffers much more variation iteration per iteration compared to other two algorithms. It seems Q Learning is still guessing around after 44th iteration as it reaches even more than 100 steps for its optimal policy.



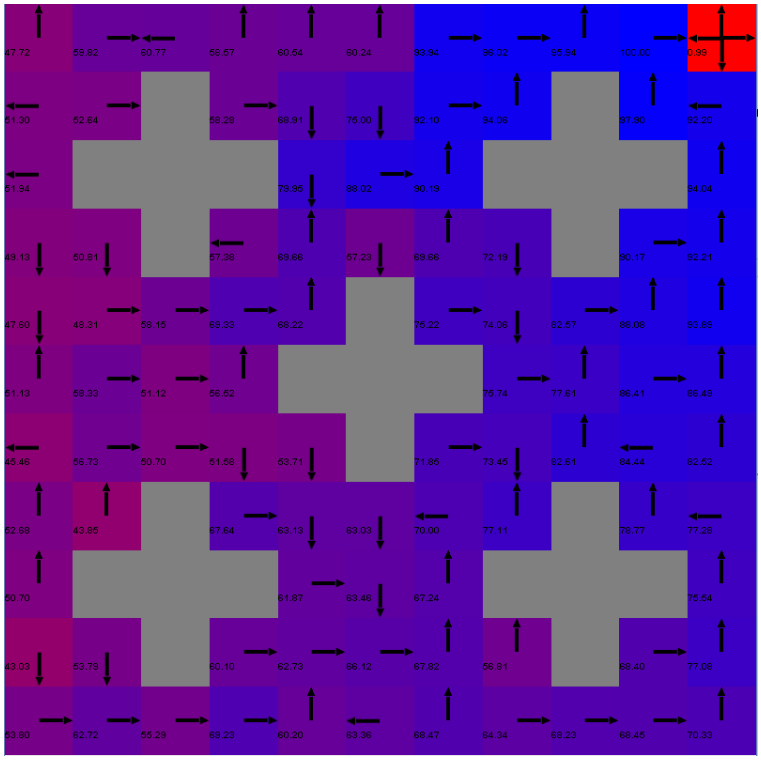


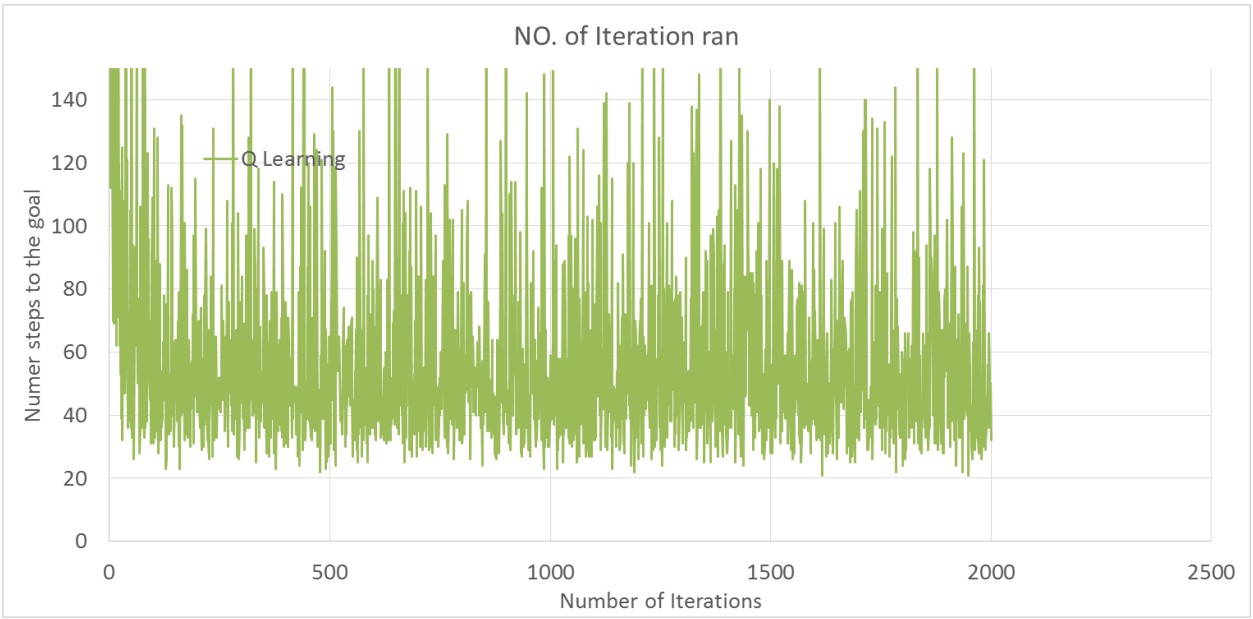
Similar to the easy version of Grid world, the processing time require to perform the experiments, is linear, regardless of which algorithms. Again the Value Iteration and Policy Iteration processing time grows linearly as iteration grows, while Policy Iteration takes longer time for each iteration than Value Iteration due to the policy generation step. Q Learning seems still holds relatively flat processing time as iteration grows.



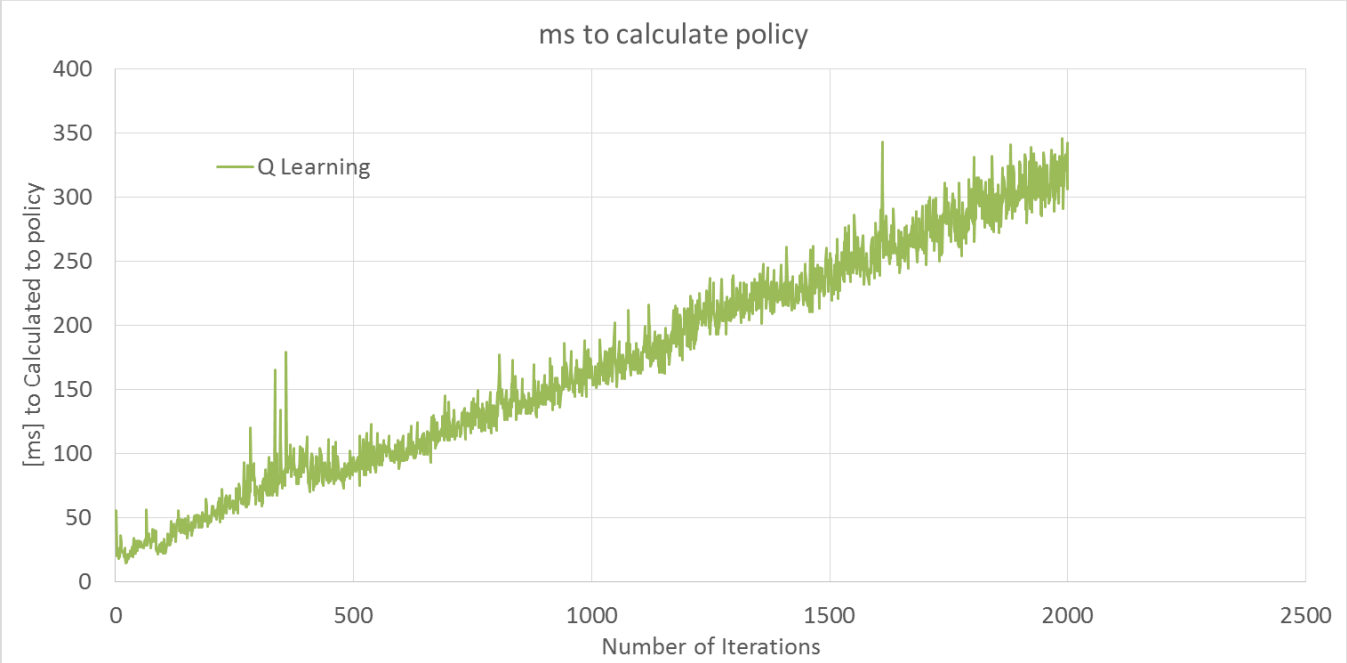
Q Learning Further Study - more iteration ran

In previous study, it seems 100 iteration is not enough for Q Learning and hence here we will try 2000 iteration to see how results of Q Learning will provide. As show below, after iteration of 200, the step required to optimal policy is much less chance to be more than 100s. Among the iteration I have ran, it is observed that it start to perform better variability than previous few iteration runs and more consistently to provide better policy. It is still sub-optimal compared to Value Iteration and Policy Iteration.





And the running time actually is an interesting observation as it starts to show linear relationship with iteration counts. When the iteration less than first 100 or so, it is possible that the agent in Q Learning is still exploring many states and sequences for the first time. As the number of iteration increase, so do the number of conflicting sequence, and for this reason, the processing time grows according like other algorithms to compute the optimal policy.



Conclusion

Given the fact that both Value Iteration and Policy Iteration require explicit transition probability and reward function pre-defined to calculate optimal policies, it is fairly unsurprising that the two algorithms converge faster to the optimal policy than Q Learning, which had to derive that information from experience. Hence it is difficult and unfair to compare Q Learning against Value Iteration and Policy Iteration since they require different level of information. It is clear that with a larger number of states, the iteration required for Q Learning grows exponentially to converge to optimal policy, while Value Iteration and Policy Iteration still holds relatively simple manner. If we are given transition probability and reward function, then we should use Value Iteration or Policy Iteration. Policy Iteration seems to compute a bit slower than Value Iteration due to its policy generation step in each iteration. If there is no transition probability and reward function defined yet, clearly we have to use Q Learning.