Markov Decision Processes

Georgia Institute of Technology CS 7641: Machine Learning Assignment Four

Xi Han

GT ID: xhan306

Introduction

This report explores three reinforcement learning algorithms value iteration, policy iteration and q learning in machine learning by implementing them to solve Markov Decision Processes (MDP) problems. Two Markov Decision Processes (MDP) problems were investigated, one with a comparatively small number of states, and the other with a larger number of situations. By comparing the performance of three algorithms, runtime, speed of convergence and total rewards in different iterations, we provide detailed advantages and disadvantages for each algorithm.

Markov Decision Processes (MDP) problems

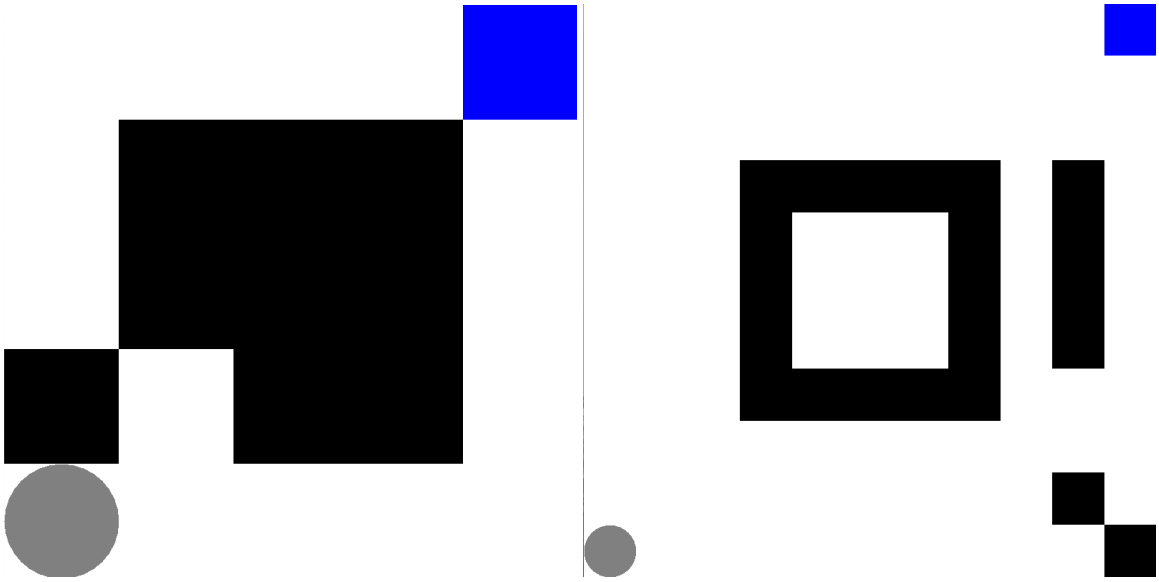
Markov decision process (MDP) deals with stochastic control process with discrete time. It provides a mathematical framework for modelling decision making in situations where outcomes are partly random and partly under the control of a decision maker. It is often used to solve optimization problems in dynamic programming and reinforcement learning and has been applied in various areas, such as robotics, automatic control, economics and manufacturing. There are five elements in a Markov decision process:

1. S: a finite set of states
2. A: a finite set of actions
3. P_a: the probability that one action in one will lead to another state
4. R_a: is the immediate reward (or expected immediate reward) received after transitioning from one state to another state
5. r: the discount factor, which represents the difference in importance between future rewards and present rewards.

The main problem of Markov Decision Processes is to find a policy to guide which actions in one state should be taken by decision makers. The policy is to maximize some cumulative function of the random rewards.

In our study, grid world problems, which are a classic problem for MDP were investigated. In the grid world, an agent starts at one grid square and moves to another grid square located elsewhere in a 2D rectangular grid of size (a, b). Reinforcement learning algorithms can be utilized to find optimal paths agents to get to the desired grid squares in the least number of moves or maximum rewards. In our implementation, the agent starts at the bottom-left grid corner (grey circle) and moves to the desired destination, the top-right grid corner (blue square). The agent can move up, down, left, right directions by 1 grid square at each step. The black squares represent the obstacles which cannot go through. The first problem is an easy grid word with a 5 * 5 grid world and the hard one is a 11* 11 grid world. The agents are trying to find the optimal policy with maximum total rewards.

This problem is very interesting because it is an abstraction of some practical applications. For example, a self-driving car need to navigate itself to arrive at the destination at the shortest paths. In the way, there are some obstacles which should be avoided. Exploring those problems can help us improving the real-world applications.



Easy Grid World Hard Grid World

Figure 1. The two grid world problems

Reinforcement Learning Algorithms

Three reinforcement learning algorithms, value iteration, policy iteration and q learning were utilized to explore those two grid world problems.

Value Iteration

Value iteration algorithm starts with a random value function and then finds a new (improved) value function iteratively until it reaches the optimal value function. And the optimal policy can be derived easily from the optimal value function. This process is based on the optimality Bellman operator and can be guaranteed to converge to the optimal values.

Policy Iteration

Policy iteration algorithm starts with a random policy, then finds the value function of that policy (policy evaluation step), then finds a new (improved) policy based on the previous value function, and so on. In this process, each policy is guaranteed to be a strict improvement over the previous one (unless it is already optimal). Policy iteration is also guaranteed to converge to the optimal policy, whereas this method often needs more calculation since it need to calculate the value function for all states by Bellman operator.

Q Learning

Q learning is a model-free algorithm compared with value iteration and policy iteration which need domain knowledge to calculate the transition function. It will discover the optimal policy and converge to its optimal value by trial and error. The Q table is initialized randomly. Then

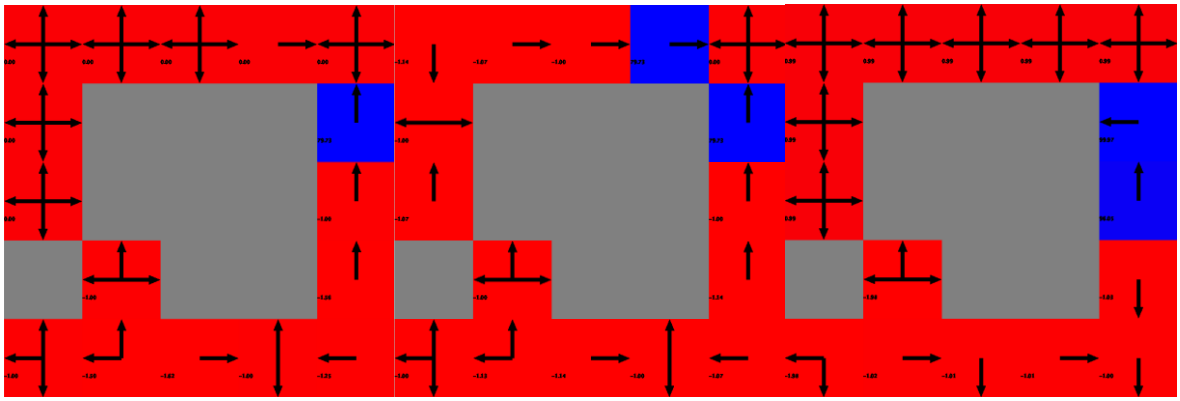
the agent starts to interact with the environment and observe the rewards of its actions, which will be recorded and used to update the estimate of Q table.

Results and Discussions

Three different algorithms were implemented with 100 iterations and exploration strategy epsilon-greedy exploration. The speed of convergence (steps taken to reach goal), computing time and total rewards were compared for three algorithms in figures.

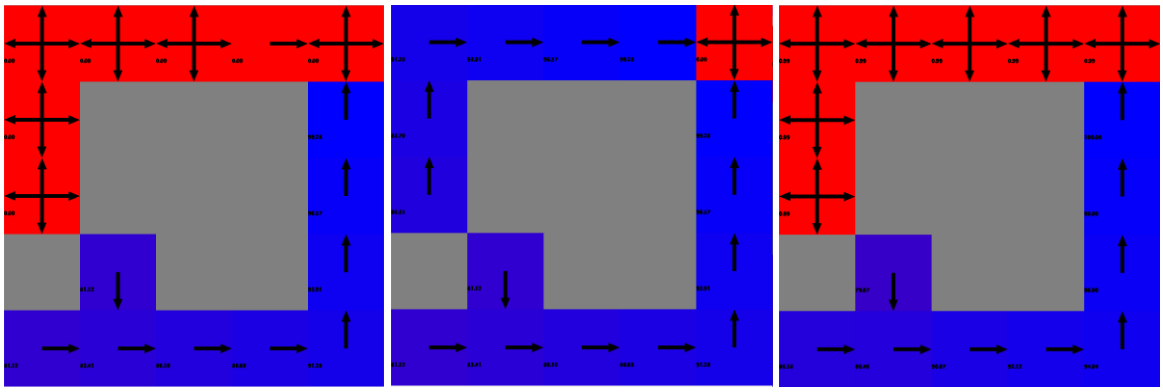
Easy Grid World

At iteration 1



Policy Iteration Value Iteration Q Learning

At iteration 100



Policy Iteration Value Iteration Q Learning

Figure 2. Optimal policies for three algorithms

The Figure 2 shows the optimal policies of three algorithms at 1 and 100 iterations for the easy grid problem. After 100 iterations, the wrong states reduce significantly for 3 algorithms. Value iteration finds the optimal policy map after 100 iterations whereas policy iteration and q learning need more iterations according to the Figure 2.

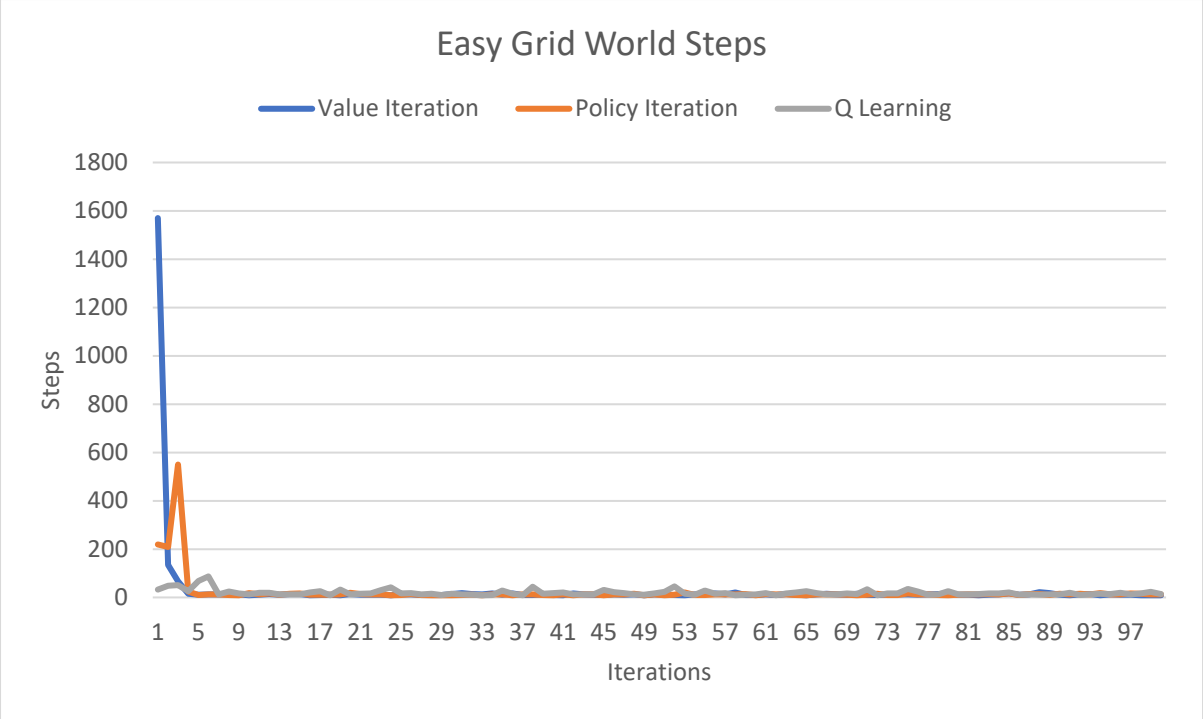


Figure 3. Easy grid world steps at each iteration

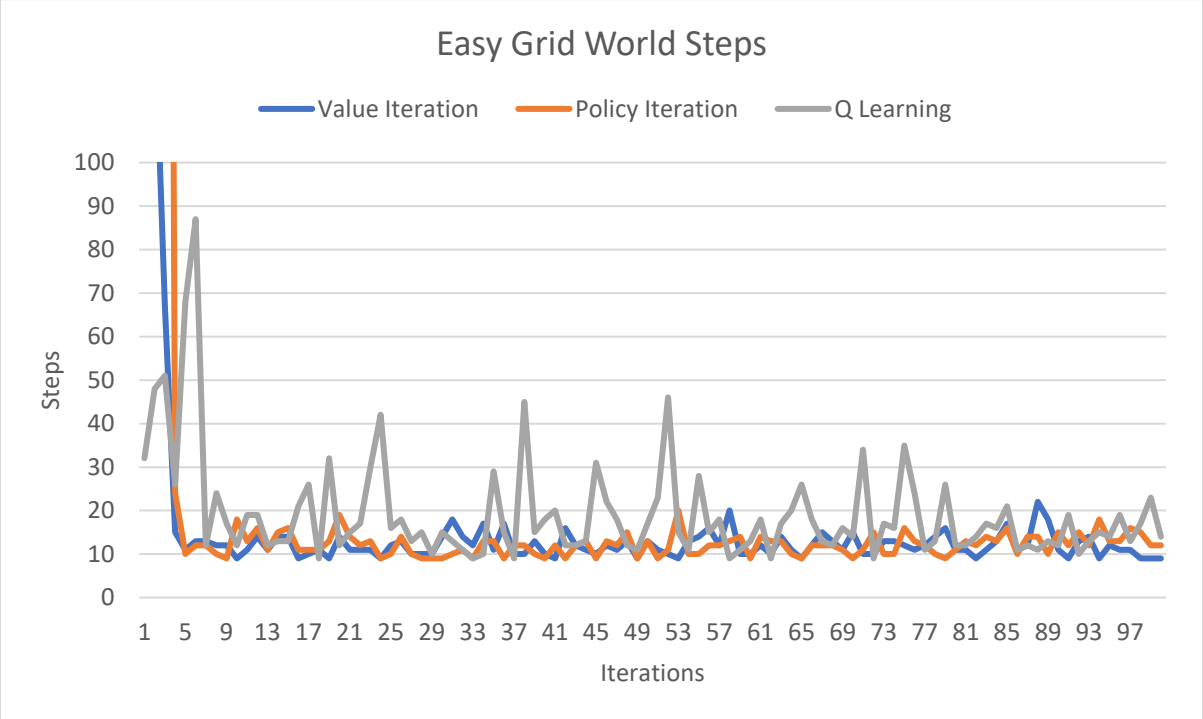


Figure 4. Easy grid world steps at each iteration (smaller range for y axis)

It can be seen from Figure 3 or Figure 4 which reduce the range of y axis in Figure 3 for observation, both value and policy iteration converge within 5 iterations, whereas q learning converges until 8 iterations.

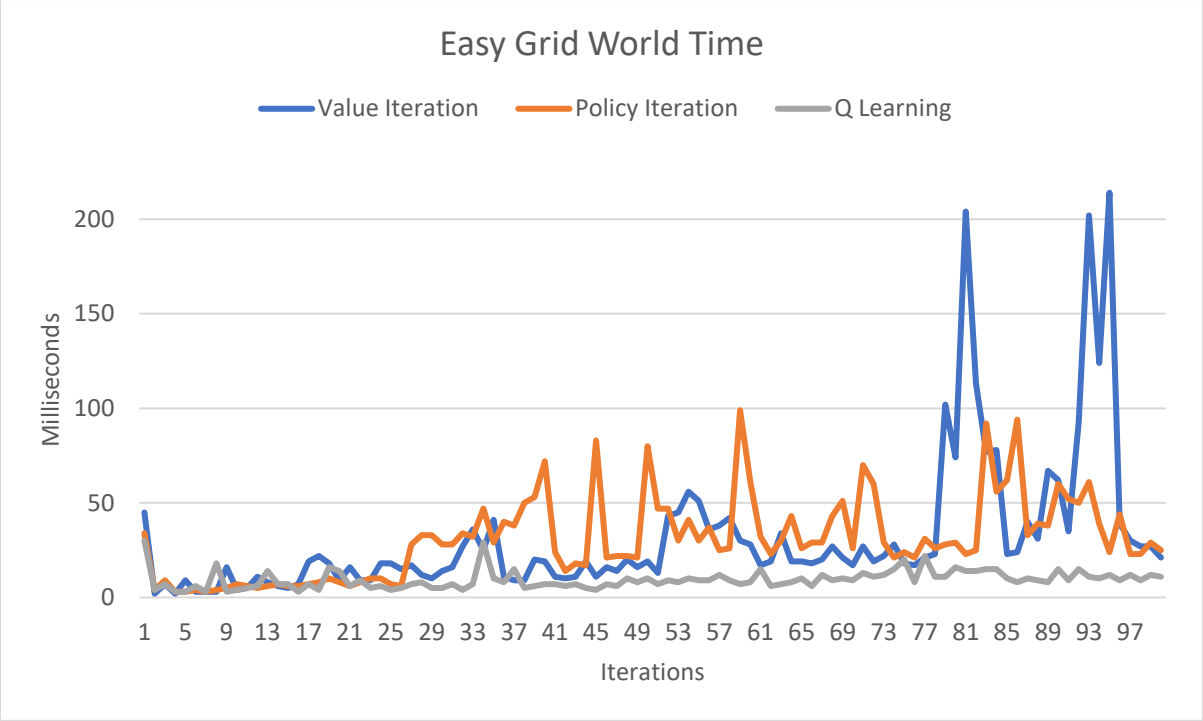


Figure 5. Easy grid world computing time at each iteration

It can be seen from Figure 5, both value iteration and policy iteration take longer time than q learning. It is reasonable as q learning just hashes its actions at a constant speed until a policy is calculated. Comparing to the others, each iteration of policy iteration is more computationally expensive since policy iteration needs to calculate the policy at each iteration although it is computationally efficient for taking considerably fewer number of iterations to converge.

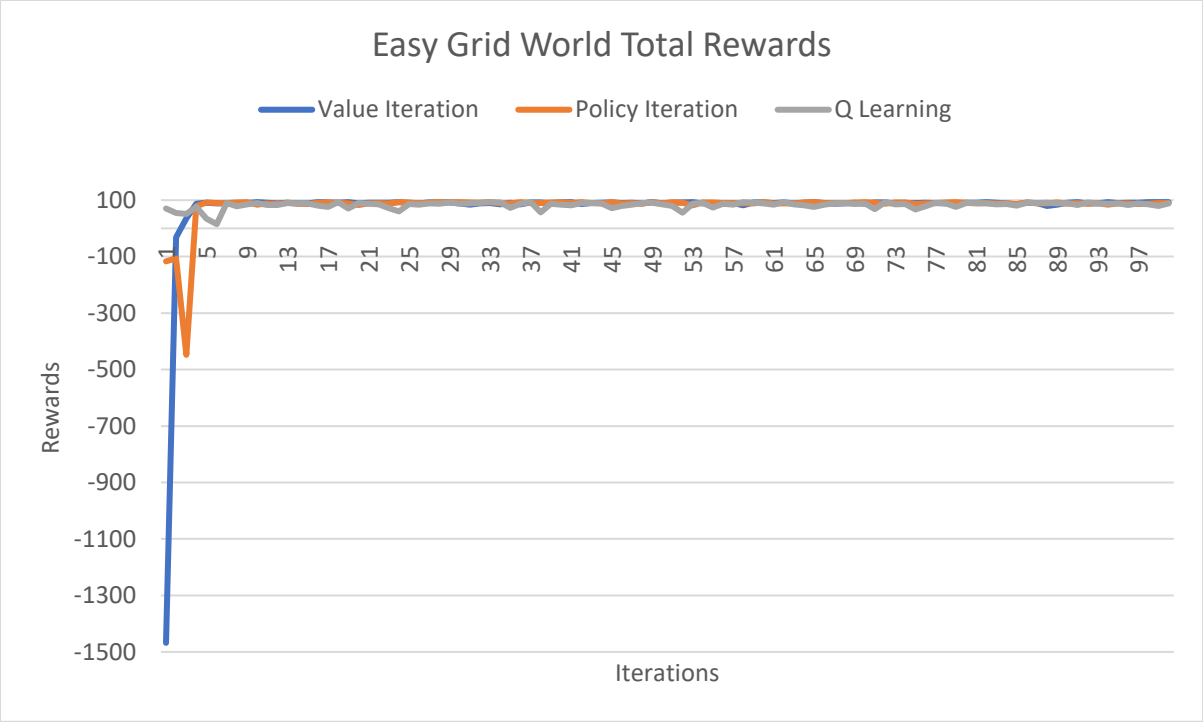


Figure 6. Easy grid world total rewards at each iteration

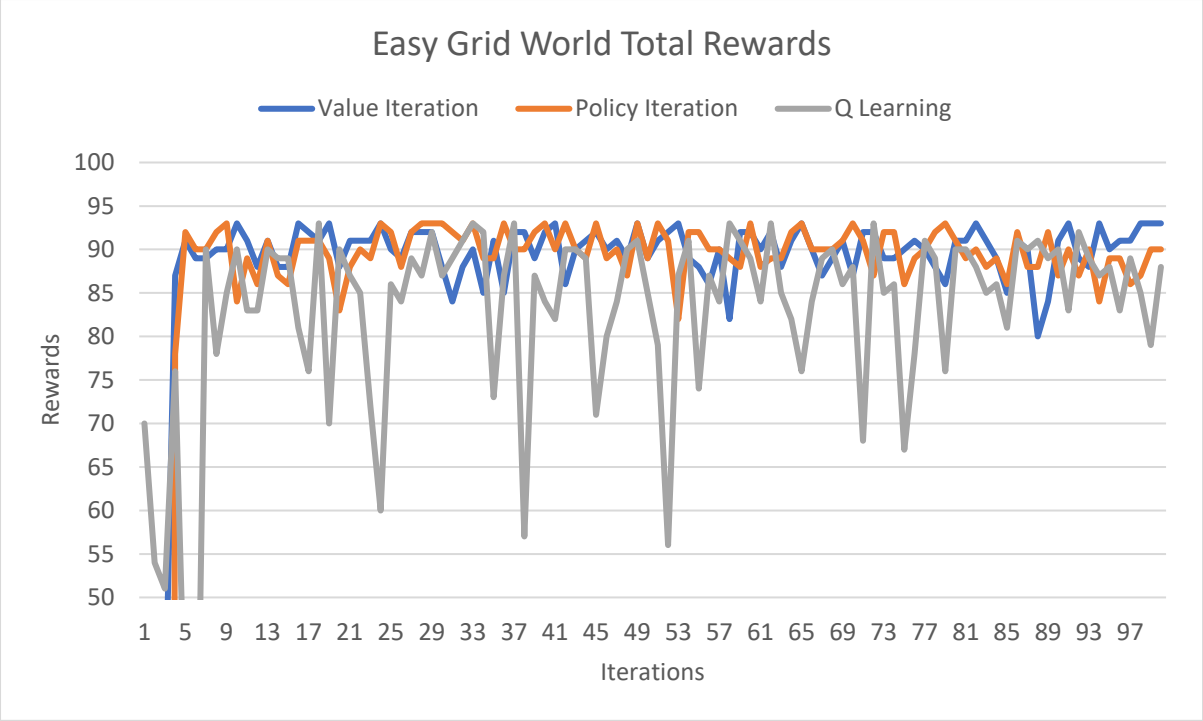


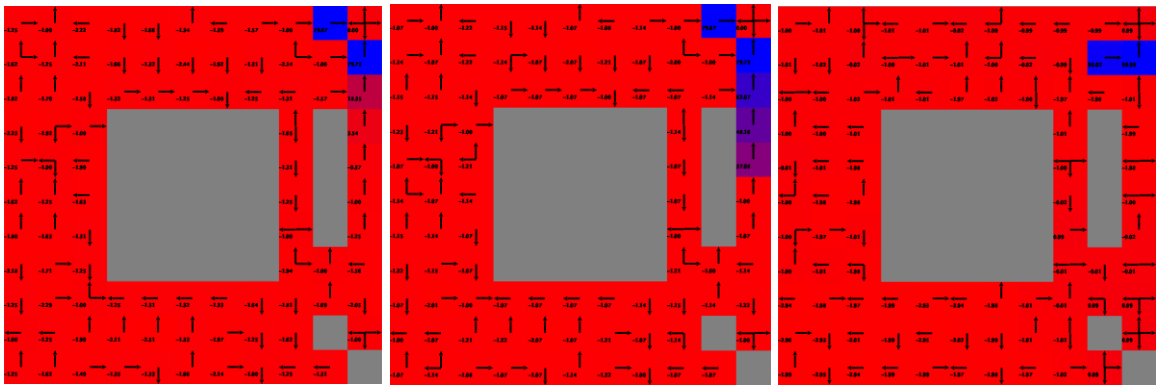
Figure 7. Easy grid world total rewards at each iteration (smaller range for y axis)

In Figure 6 and Figure 7 which narrows down the range of y axis of Figure 6 for observation, it can be observed that both value iteration and policy iteration converge to about 90 within 4 iterations quickly, whereas q learning fluctuates until 9 iterations to reach about 90 range. In addition, the q learning has larger variations of total rewards even after 9 iterations.

Hard Grid World

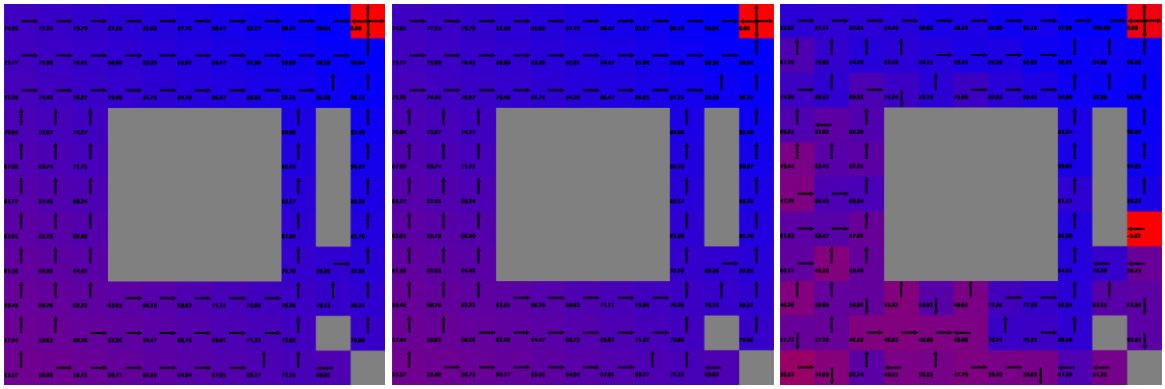
Figure 8 shows the optimal policies for three algorithms at 1 and 100 iterations. It is obvious that both value iteration and policy iteration find the optimal policy maps after 100 iterations compared with q learning where there are still some wrong states. Q learning needs more iterations to explore the right policy. After 1000 iterations, q learning corrects some states compared with 100 iterations. However, q learning retains some of confusion in the grid world and needs further improvement.

At iteration 1



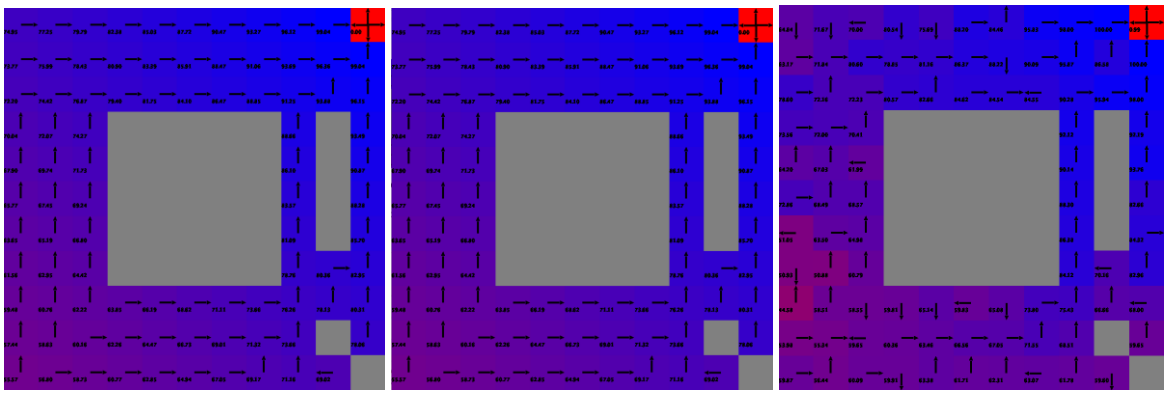
Policy Iteration Value Iteration Q Learning

At iteration 100



Policy Iteration Value Iteration Q Learning

At iteration 1000



Policy Iteration Value Iteration Q Learning

Figure 8. Optimal policies for three algorithms

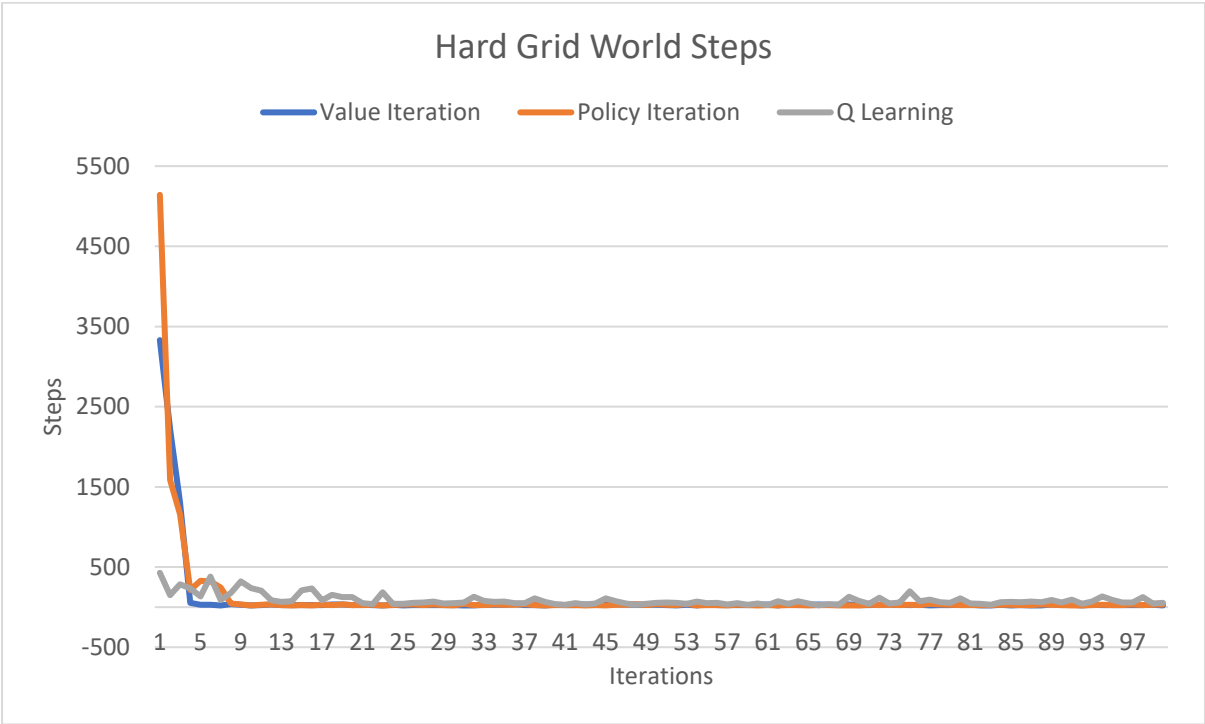


Figure 9. Hard grid world steps at each iteration

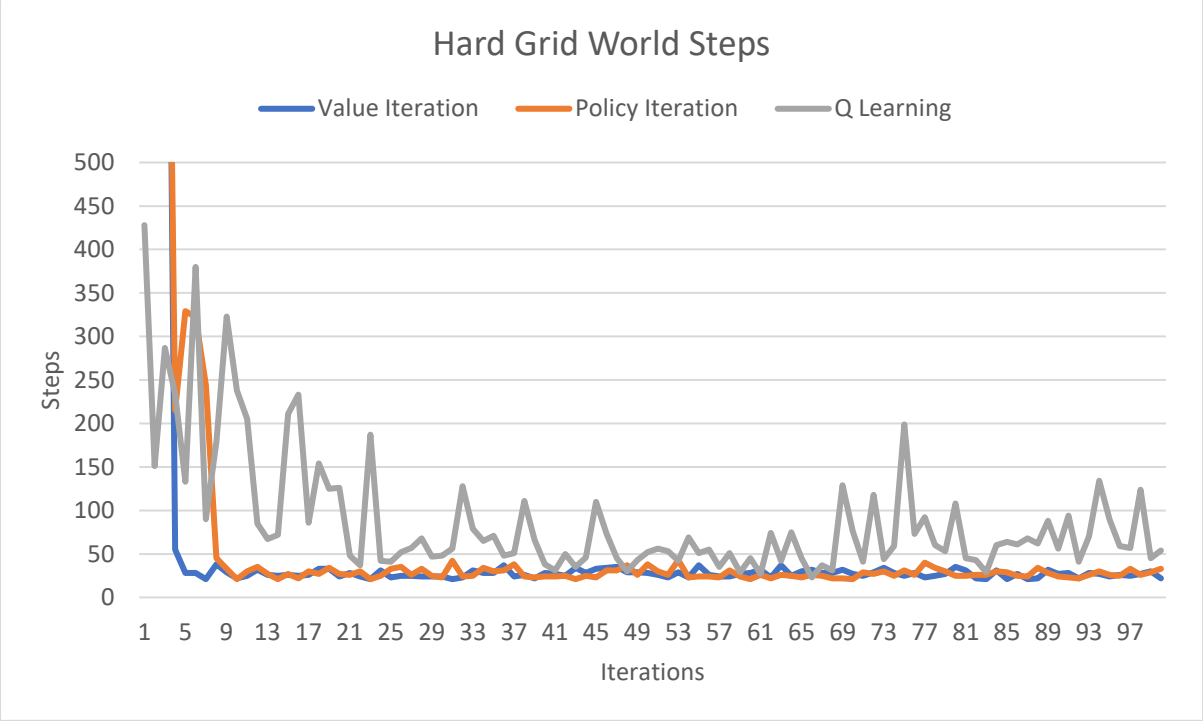


Figure 10. Hard grid world steps at each iteration (smaller range for y axis)

Figure 9 and 10 show the same trend with the easy grid world problem. Both value iteration and policy iteration converge to 30 within 9 iterations, while q learning fluctuates for all the 100 iterations.

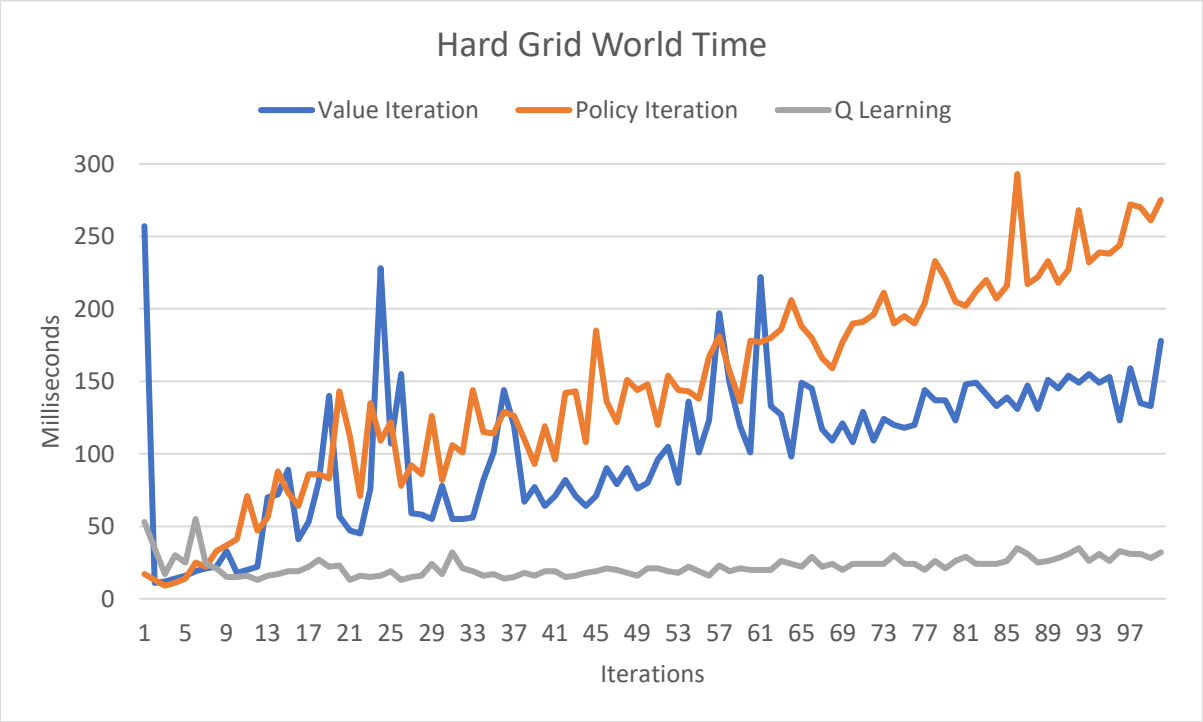


Figure 11. Hard grid world computing time at each iteration

In Figure 11, Q learning has a constant computing time since it only need hash value access. Policy iteration costs longest time among the three algorithms as it needs to compute the policy from value function at each iteration.

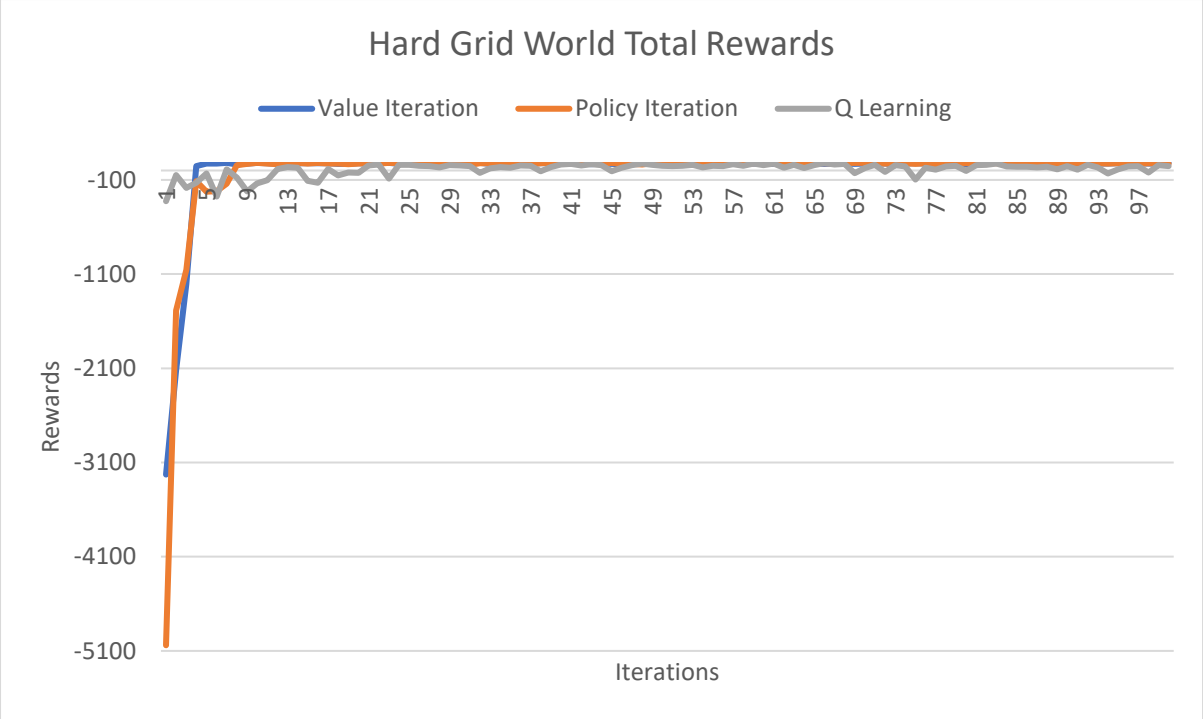


Figure 12. Hard grid world total rewards at each iteration

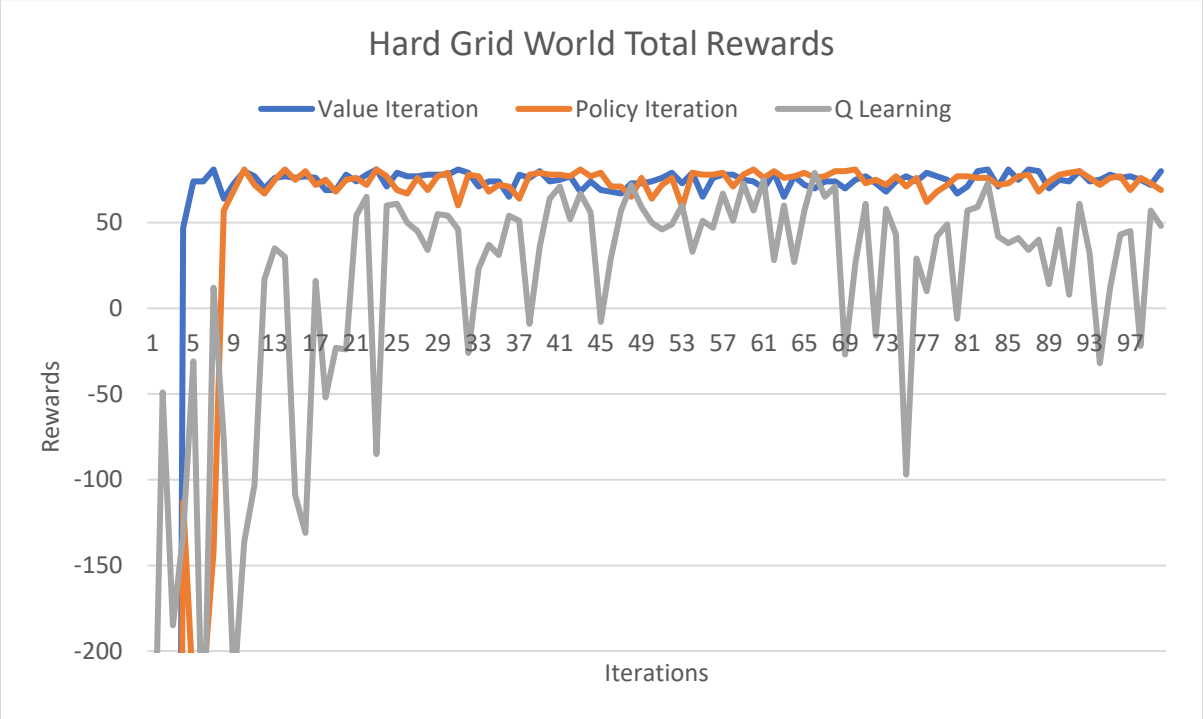


Figure 13. Hard grid world total rewards at each iteration (smaller range for y axis)

In Figure 12 and 13, both value iteration and policy iteration converge to 80 quickly within 9 iterations, whereas q learning has a very large variance for all the 100 iterations. More iterations may be needed for the convergence of q learning. Therefore, 1000 iterations were implemented and shown in Figure 14. It can be seen that q learning has not converged until 1000 iterations although the variance is reducing with iterations.

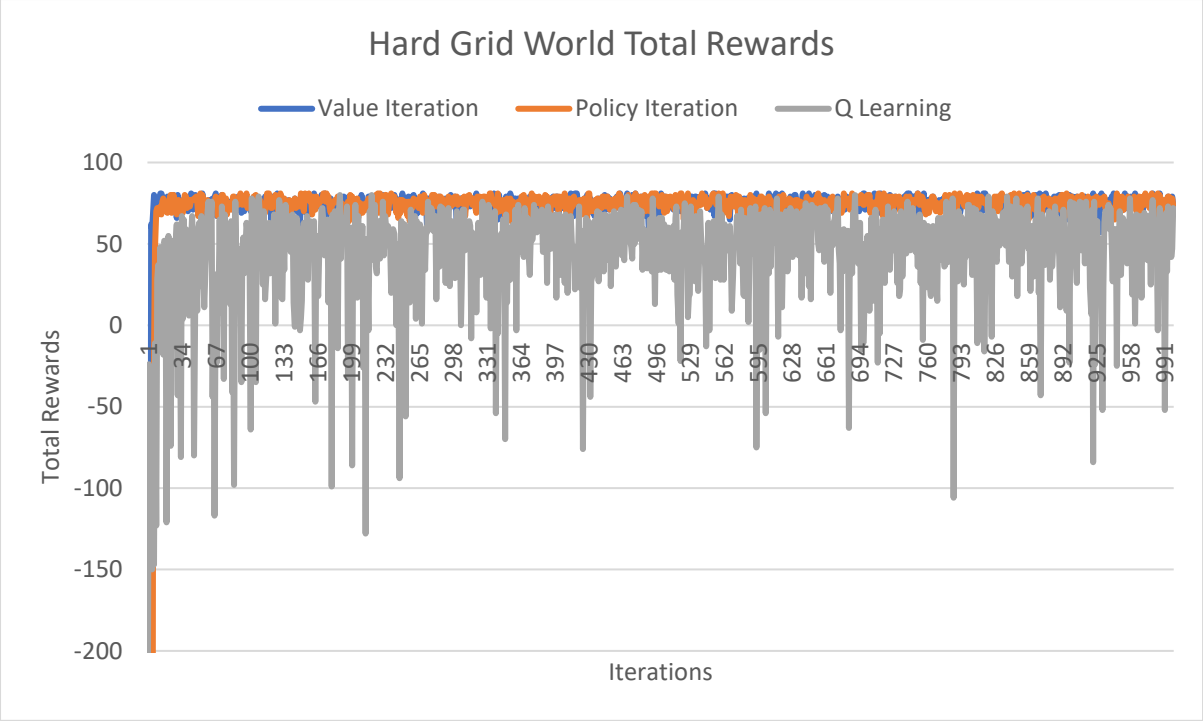


Figure 14. Hard grid world total rewards for 1000 iterations

Conclusion

According to the analysis above, value iteration and policy iteration have better performance compared with q learning for two grid world problems. It is because value iteration and policy iteration are based on given domain knowledge, whereas q learning has to explore and learn from the environment by trial and error and often faces dilemma. Therefore, q learning needs more time compared with another two algorithms and if we already have domain information, value iteration and policy iteration should be chosen to solve the problem.

In theory, value iteration and policy iteration outperform q learning with enough domain knowledge, whereas in real-world problems, q learning is more practical since there is less domain knowledge we can get for value iteration and policy iteration. In that case, q learning would be more adaptable than value iteration and policy iteration. In addition, in those real-world problems, an optimal policy may be calculated with value iteration or policy iteration from the domain knowledge learned by q learning.