HW4 Report
Markov Decision Processes

Subject :
CS 7641 Spring 2017 - OMS

Author :
Melisande Zonta Roudes
GT account name : mzr3

April 24, 2017

1 Introduction

The aim of this assignment is to study some techniques of the reinforcement learning in order to analyze work of an agent from a machine learning perspective. We will consider two different problems since the first problem will consist of a small number of states and the other has a large number of states. The analysis is performed according to two planning algorithms and one learning algorithm. This assignment was implemented thanks to the Burlap library and the curves created on R. The algorithms chosen are Value Iteration, Policy Iteration and Q Learning.

2 Used Markov Decision Process Problems

2.1 Description

One of the simplest algorithm and classic problem in Markov Decision Processes called Grid World. The environment consists of an agent (pink circle), walls (black squares) and the goal which is the final state (red square). The agent lives in a grid, walls block the agent's path. The aim of this Markov Decision Problem is that the agent has to reach the final state in the less number of steps possible by traversing the grid. The agent's actions do not always go as planned : 80% of the time the action North takes the agent North if there is no obstacle, 10% of the time North take the agent West and 10% East. If there is a wall in the direction the agent would have been taken, the agent stays put. the agent receives rewards each time step : small "living" reward each step (they can be negative as well). In any cases, big rewards come at the end. In order to visualize the process, two different grids were designed with different number of states.

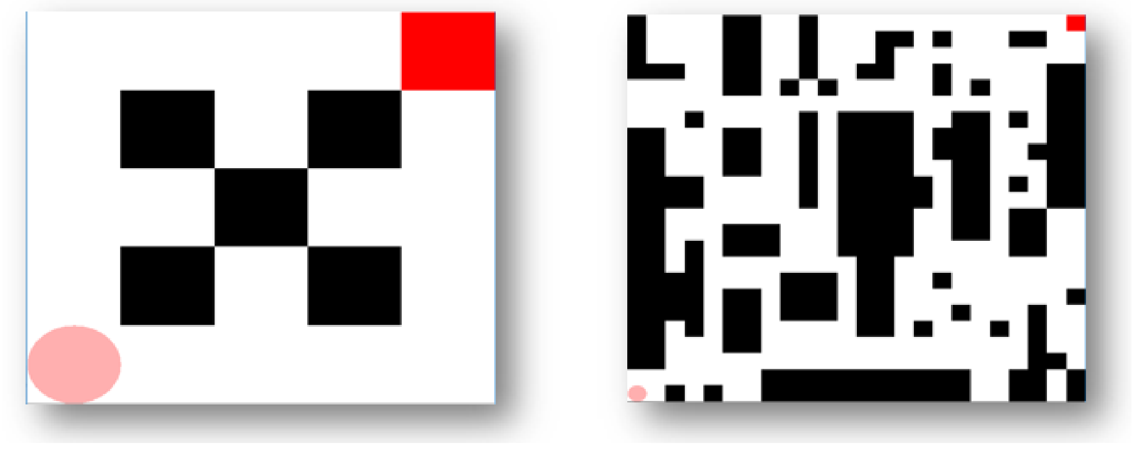


Figure 1: Grids
a. Grid World : Low difficulty. *b.* Grid World : High difficulty.

The easy grid has only 20 possible states and only one obstruction between the agent and the goal. The hard grid has 331 possible states and many obstacles between the agent and the target.

The way of reaching the terminal state in the least number of steps is by the computation of the reward function, or the cost of each future action. For Grid World problems, each action that does not allow to reach the terminal state leads the agent to loose 1 point for each action. In deterministic single-agent search problem, want an optimal plan, or sequence of actions, from start to a goal. The actions must also have probability of success and failure. Indeed a transition matrix is built that determines the probability of success that the agent has to move successfully in the direction intended (south, north, east, south, west).

2.2 Interest

Though the Grid World problem seems to be a simple and classical problem, it is really a real world problem if we consider all the robotics applications. Indeed, we hear about all the improvement done in the development of automated vehicle which has to adapt itself to its environments, deal with new obstacles and reach a goal. This situation can be compared to the development of an artificial intelligence required for a self driving vehicle. The problem is the same since the agent (the vehicle) has a starting point and a destination, each of the map that surrounds the car and leads it to the destination could be considered as the states of our grid. The obstruction are the buildings or the pedestrian.

By analogy, the agent would have to decide if he has to accelerate or decelerate (overall the speed), and the direction he has to take to reach the following state (angle of the wheels). The agent would have also to compute the probabilities of success in switching stated due to obstacle in front of the vehicle. In our Grid World, the agent has to take 1 of 4 possible actions and has to compensate for its likelihood of success.

3 Algorithms

3.1 Value Iteration

Value iteration is a planning algorithm allowing to compute an optimal MDP policy and its value. It is an iterative procedure that calculates the expected utility of each state using the utilities of the neighboring states until the utilities calculated on two successive steps are close enough with a predefined threshold value. The smaller the threshold is, the higher is the precision of the algorithm. Given certain conditions on the environment, the utility values are guaranteed to converge. The process of propagating backwards is never ending so an arbitrary end point is chosen when the algorithm approximates well enough. The stopping thresholds that can be used are maximum number of iterations and maximum delta threshold (when maximum change in the value function is less than this value the algorithm will terminate).

3.2 Policy Iteration

The policy often becomes exactly optimal long before the utility estimates have converged to their correct values. It leads to the finding of optimal policies, called policy iteration. This method picks an initial policy, usually just by taking rewards on states as their utilities and computing a policy according to the maximum expected utility principle. Then it iteratively performs two steps : value determination, which calculates the utility of each state given the current policy, and policy improvement, which updates the current policy if any improvement is possible. Policy iteration tries to computes the optimal policy via a two step process. Step one is the inner value iteration which is performed first followed by step two which the policy iteration. The policy iteration is performed until it converges to the best possible solution and then this two step process is repeated until it completely converges.

3.3 Q-Learning

The learning algorithm of choice is Q-Learning which is a reinforcement learning technique. It can be used to find an optimal action-selection policy for any given Markov Decision Process. It starts by learning an action-value function that gives an expected utility by taking that action when in a particular state and following an optimal policy from that point onward. The main advantage of this algorithm is that it is able to make a comparison between the expected utilities for the available actions without even requiring a model of the environment.

4 Easy Grid World

4.1 Visualization of the process

In order to see the differences of the three algorithms, we test them on the low level Grid World. As the number of iterations is an important factor in the convergence, we will launch our map on one iteration, then on 100 iterations.

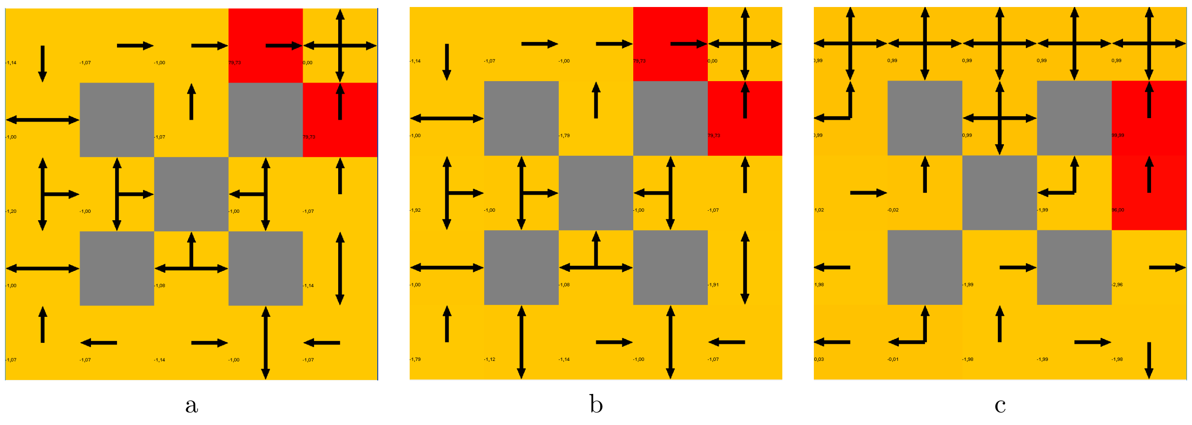


Figure 2: Easy Grid World : One iteration
a. Value Iteration. *b.* Policy Iteration. *c.* Q-Learning

The curves on the figure 7 represent the policies generated by the three algorithms after running only a single iteration. The Value Iteration and Policy Iteration seem to produce very similar policies with one exception. The result is astonishing since the policies generated by those algorithms are quite good and show movement towards the final state. However, the policies computed with Q Learning suffers from many policies that move the agent directly towards walls, which shows that it is really not optimal. We can see that the initial state which is in yellow become redder by getting closer to the goal, because the cumulative rewards are increasing. The final state remains in yellow because once the final state reached, it has not to leave this state.

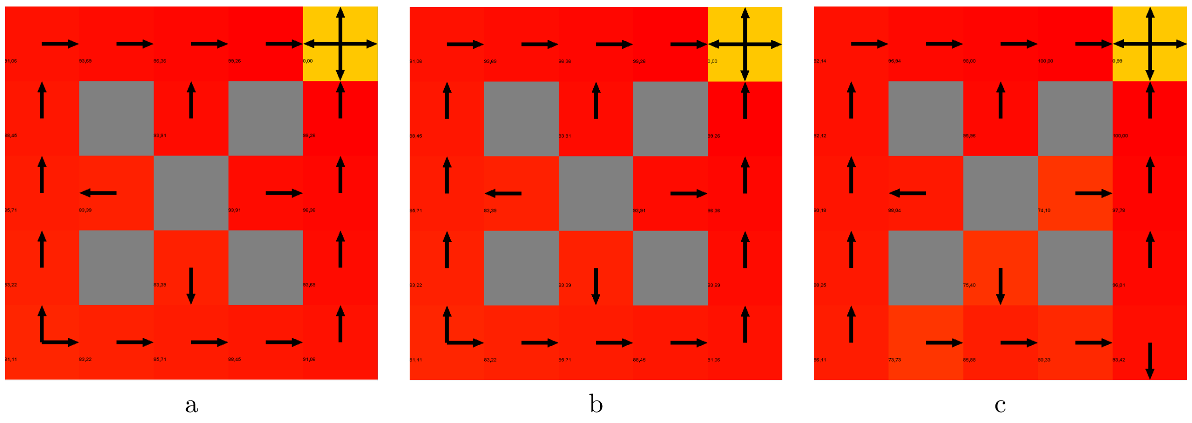


Figure 3: Easy Grid World : 100 iterations
a. Value Iteration. *b.* Policy Iteration. *c.* Q-Learning

Now we want to provide the same experiments for 100 iterations, the policies change completely. We can observe that Value Iteration and Policy Iteration converge on the optimal policy for an initial action moving towards North or East. The improvement can be seen especially on the behaviour of our Q-Learner since he was able o find the optimal policy if the initial action was to move north. By looking closely at the Q Learning policy, we notice that if the agent moves as its initial action towards east, the policy can't reach the goal. We see in fact that all the scenarii lead to failure except for the movement towards North as initial step. The Q Learning policy is less optimal than the two others. The grid is completely in red except for the final state which is logical since with 100 iterations, we reached the optimal policy.

4.2 Application of the algorithms

By applying the algorithms, we want to graphically demonstrate the convergence of the policies, three plots are generated for each algorithm and then a comparison is made with the three curves. Each algorithm is run over 100 iterations, a policy is then generated and a simulation run using that policy in order to calculate the number of steps required to reach the goal state. On the graphs, we can see that Value Iteration and Policy Iteration converge towards the optimal policy in 5-7 iterations. In comparison, Q Learning do not seem to converge until 20^{th} iteration and even then the number of steps to reach the goal is still random with spikes along the iterations.

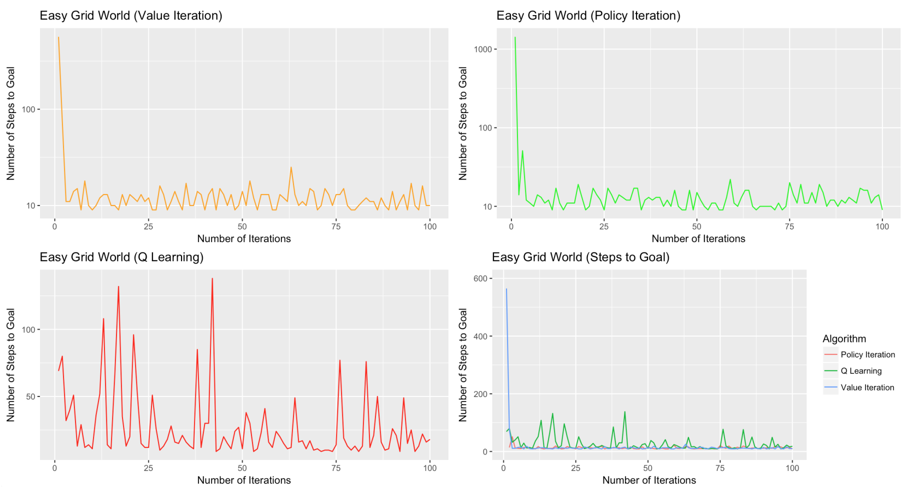


Figure 4: Easy Grid World : Steps to Goal

Another parameter interesting to compare the three algorithms is the number of miliseconds required for the algorithms to run and generate a policy along the increment of the iterations. The three algorithms appear to be running in linear time. Policy Iteration clearly takes the longest time to run, because of the policy generation step that occurs during each iteration. Value Iteration which does not run a little bit faster than Policy Iteration. Q-Learning appears to run at a constant rate of 10-15 miliseconds regardless of the number of iterations.The speed at which the algorithm runs is probably due to the fact that the algorithm, throughout the series of iterations, is not actually performing any computation but rather simply hashing the actions taken and the rewards achieved until a policy is calculated at the very end.

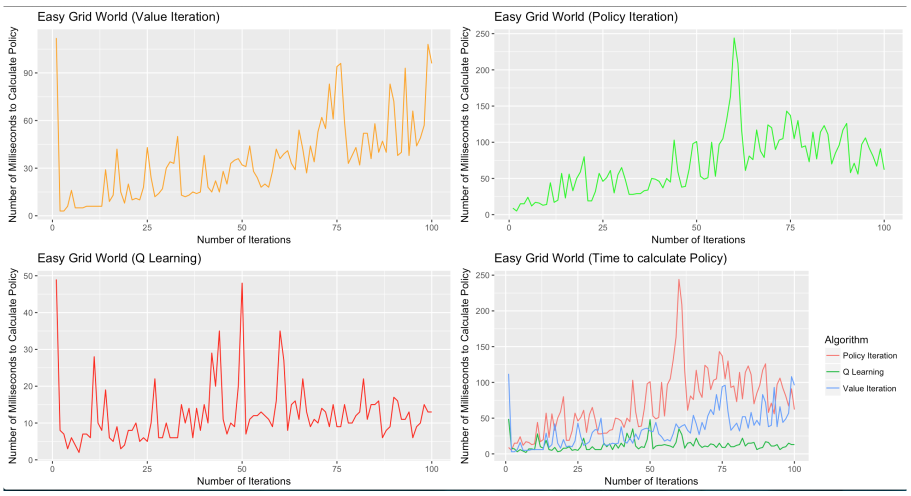


Figure 5: Easy Grid World : Time to Calculate Policy

We also plotted the number of rewards gained for the optimal policy, as shown in Figure 5, and we can clearly observe that It is highly correlated to the number of steps to Goal. Here again, Value Iteration and Policy Iteration converge after only 4-7 iterations whereas Q Learning converges after 40 iterations and still have. occasional spikes even as the number of iterations increases.

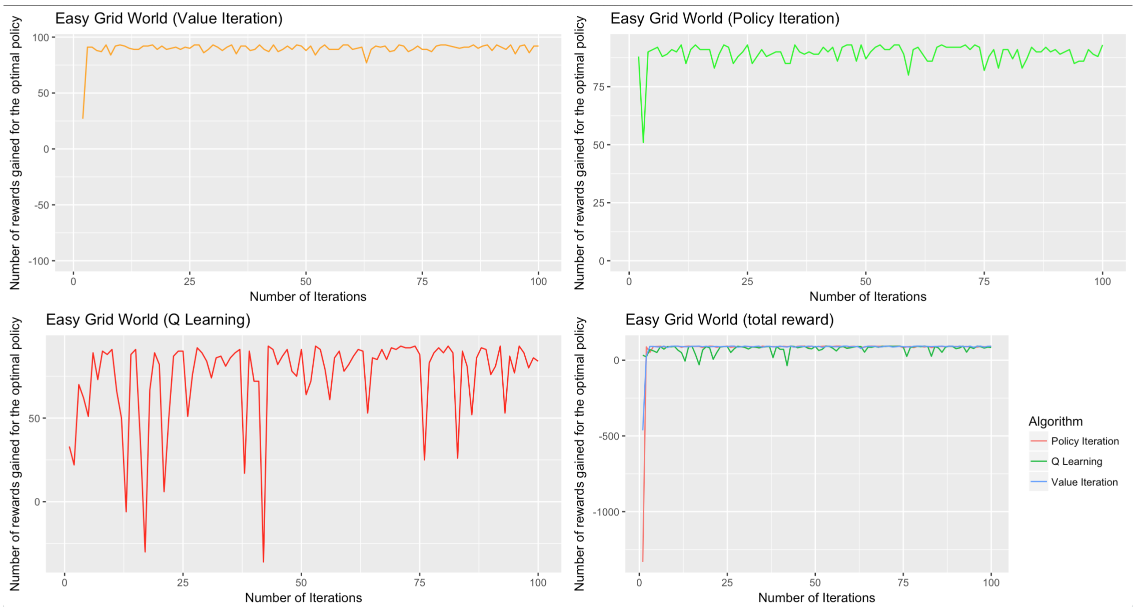
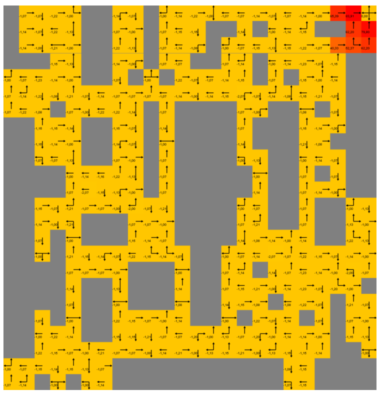


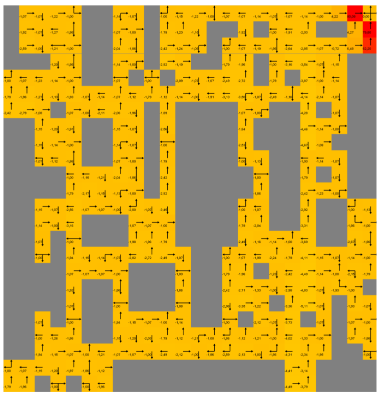
Figure 6: Easy Grid World : Total rewards

5 Hard Grid World

The first grid was not enough complex to demonstrate the efficiency's differences between algorithms so we create a bigger map with more obstructions. The number of possible states is now 331 as we have a grid of 24 × 24 and a lot of obstacles.

5.1 Visualization of the process


a


b

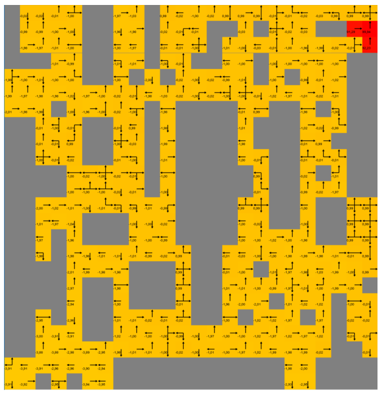
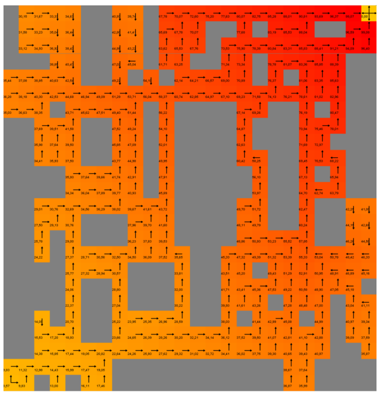
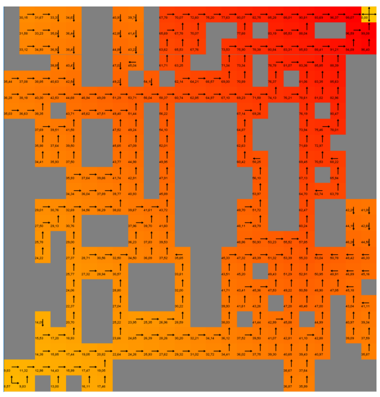

c

Figure 7: Hard Grid World : One iteration
a. Value Iteration. *b.* Policy Iteration. *c.* Q-Learning

The difficulty can be immediately observed after generating the policies computed for one iteration. Here the policies are close to random. Value Iteration and Policy Iteration provide some guidance when the agent is within 4 space from the goal state. Q-Learning perform really badly providing no guidance at all unless the agent is within 1-2 spaces from the final state. The grid remains in yellow until as we said 2 steps before the goal state for Value Iteration and Policy Iteration.


a


b

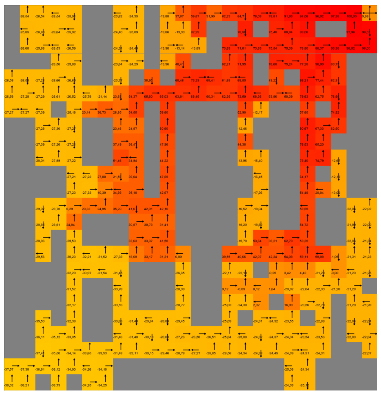

c

Figure 8: Hard Grid World : 100 iterations
a. Value Iteration. *b.* Policy Iteration. *c.* Q-Learning

As before, we increase the number of iterations in order to obtain a better policy. Again, Value Iteration and Policy Iteration have almost the same policies and converged towards the optimal policy. Unlikely to the easy grid world, the Q-Learning algorithm hasn't improved so much since many arrows are still pointing in another direction than the one of the goal state. Moreover, on this grid, we can observe that visiting all the states is a challenge that can be reached only by increasing the number of iterations. Again, the more we get close to the final state, the darker the grid is. We observe that concerning Value Iteration and Policy Iteration, it's getting gradually darker so the rewards are improving little by little, unlikely Q Learning becomes all that once red.

5.2 Application of the algorithms

Again, we want to compare the three algorithms' abilities to converge to an optimal policy, we graphed the number of steps required to reach the goal given the policy calculated over 100 iterations. From this graph, it is clear that Value Iteration converges on optimal policy by the 23rd iteration and Policy Iteration converges on optimal policies by the 25th iteration averaging 50 steps to the goal. Q Learning, on the other hand, appears to converge by the 75th iteration averaging 205 steps to reach the goal.

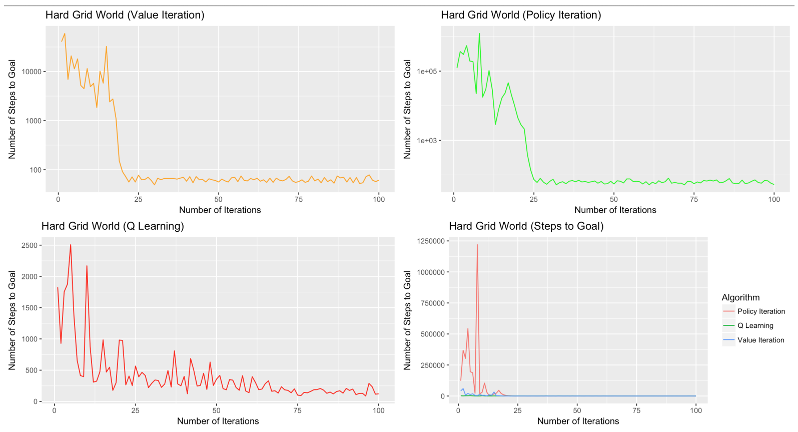


Figure 9: Hard Grid World : Steps to Goal

Much like in the low difficulty Grid World, the time required to perform the experiments, regardless of the algorithm, is linear. Policy Iteration is still the slowest algorithm with Value Iteration as the second slowest. Q Learning, also follows a similar pattern as before where throughout the experiment, the time to generate a policy is almost constant regardless of the number of iterations.

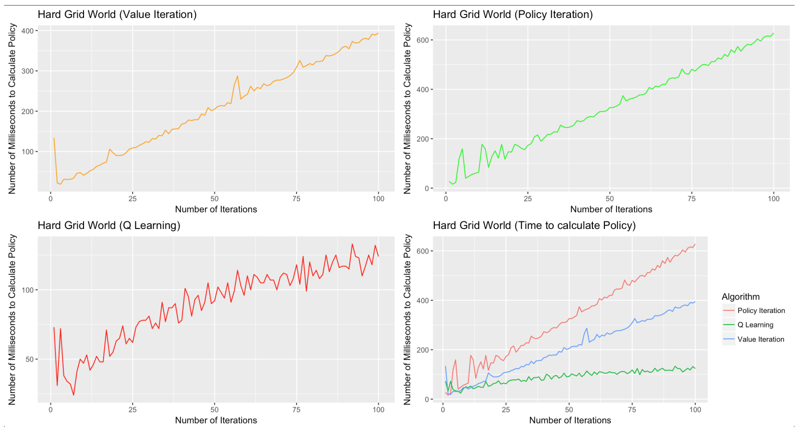


Figure 10: Hard Grid World : Time to Calculate Policy

We also plotted the number of rewards gained for the optimal policy, as shown in Figure 9. Here again, Value Iteration and Policy Iteration converge after only 23 iterations whereas Q Learning almost converges after 75 iterations and still have. occasional spikes even as the number of iterations increases.

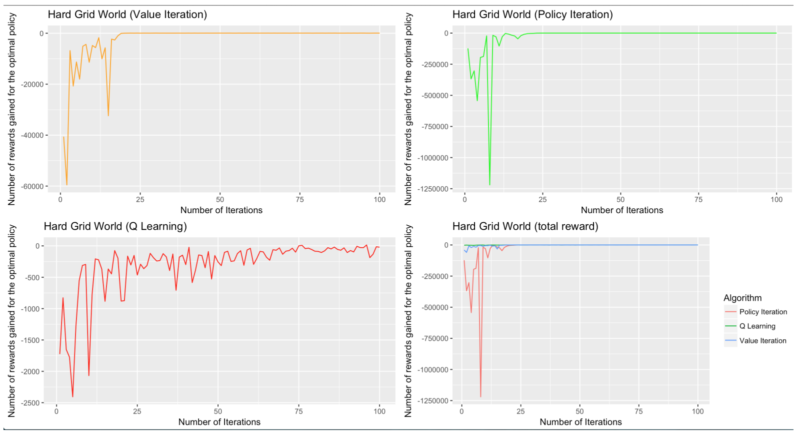


Figure 11: Hard Grid World : Total rewards

6 Comparison of the two first algorithms

We wanted to see how our three algorithms behave over the two different difficulties of the Grid World Problem. So we plotted their number of steps to Goal, number of milliseconds to calculate policy and their number of reward for the optimal policy as function of the number of iterations. We can observe that both the algorithms need more iteration to converge to an optimal policy for the hard problem, as they both converge by the 4-7th iterations for the easy problem and converge by the 23-25th iteration for the hard one. Moreover, the time to calculate the optimal policy is linear no matter the difficulty of the problem, however the slope is getting greater with the increase of difficulty.

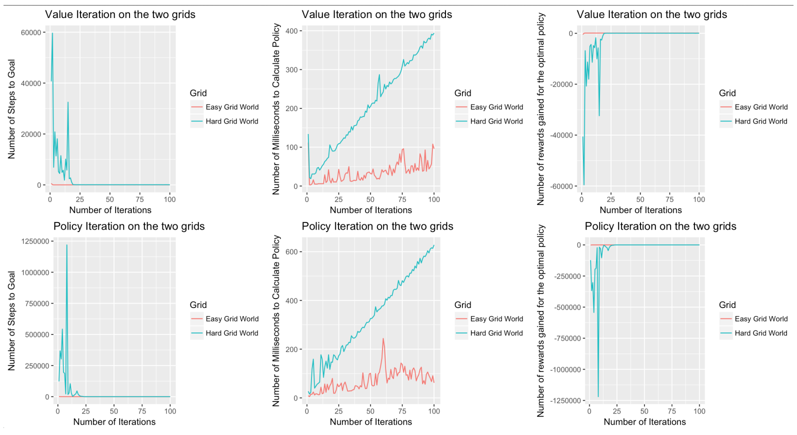


Figure 12: Comparison between the two grids

7 Deeper analysis of the Q-Learning algorithm

In the previous section, when we compared the three algorithms head-to-head on the hard Grid World, it was clear that Q Learning was not able to extract an optimal policy when given 100 iterations. In this section, we tried to increase the number of iterations to 10000 and see if the Q Learning algorithm converges on an optimal policy or at least guarantees some level of convergence. The policy map that resulted from this experiments can be seen below. Unlike the policy map extracted in the previous section, when Q Learning was only run 100 times, this time the policy looks a lot more decisive about the directions the agent should move from the starting location all the way to the goal. While some of the policies for several states are counter-productive in achieving the goal, generally speaking it is a much more attractive policy.

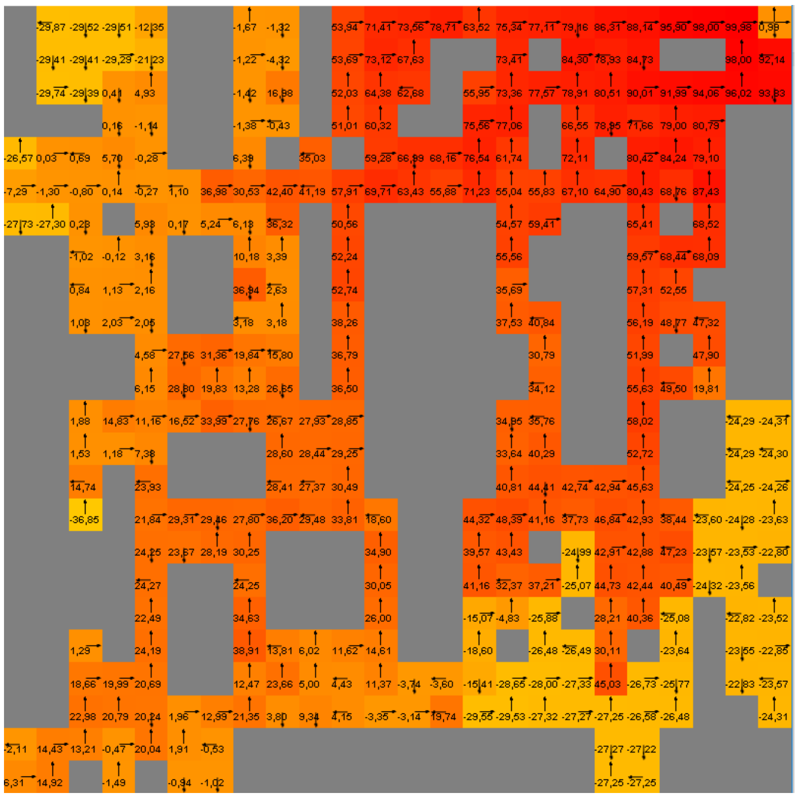


Figure 13: Hard Grid World over 10000 iterations

8 Conclusion

Value Iteration and Policy Iteration converged to the same optimal policy in the same number of iterations. While Value Iteration and Policy Iteration are given a lot of information about the process upfront, Q Learning must derive this information from experimentation by interacting with the process. This ultimately translates to a more difficult problem that requires more iterations and time in order to converge to an optimal policy like the ones calculated by Value Iteration and Policy Iteration. In the easy Grid World with only 20 states, Value Iteration and Policy Iteration were able to converge to an optimal policy in only 4-7 iterations while Q Learning required around 40 iterations in order to achieve a similar policy. In the hard Grid World, where there were 331 possible states, there was a magnification of the differences between the algorithms' abilities to converge on optimal policies. While Value Iteration and Policy Iteration were able to converge on optimal policies by the 23rd to 25th iteration, Q Learning did not come close to converging on an optimal policy even after 10000 iteration as the optimal policy given by the algorithm still has a lot of failures.

In conclusion, it's difficult to compare these algorithms over two inherently different problems. While it is clear that with a higher number of states, there is an exponential increase in number of iterations required for Q Learning to converge on an optimal policy.