Project 4 – Markov Decision Processes

Introduction: A Markov Decision Process (MDP) represents a set of states, actions and transitions in which the future states of the process only depend on its current state. The objective of this project is to explore, compare and contrast the characteristics of **Policy Iteration (PI)**, **Value Iteration (VI)**, and a **Reinforcement Learning (RL)** technique known as **QLearning (QL)**, by using them to solve various MDPs.

Rationale behind Choice of MDPs: The project brief requires that 2 MDPs (1 "easy" and 1 "hard") be created and solved using the techniques listed above. I decided to create 2 grid world problems, an **Easy Grid with 11x11 states**, and a **Hard Grid with 22x22 states**. The objective in grid worlds is for an agent to navigate the grid world from a start state to a destination/terminal state as efficiently as possible.

I decided to use grid world for a number of reasons. First, grid worlds are well known, commonly used, and widely understood. Also, grid worlds are relatively easy to solve, and do not introduce any unnecessary extra complexity to the experiments, whilst still allowing all relevant characteristics of the various RL techniques to be easily observed.

Finally, and equally as important, grid worlds are supported out-of-the-box by my choice of implementation methodology, which is discussed further in the section below.

Figure 1 below shows the easy and hard grids used in these experiments. The grey circles represent the start position, the blue boxes represent the destination, and the black regions represent walls/barriers.

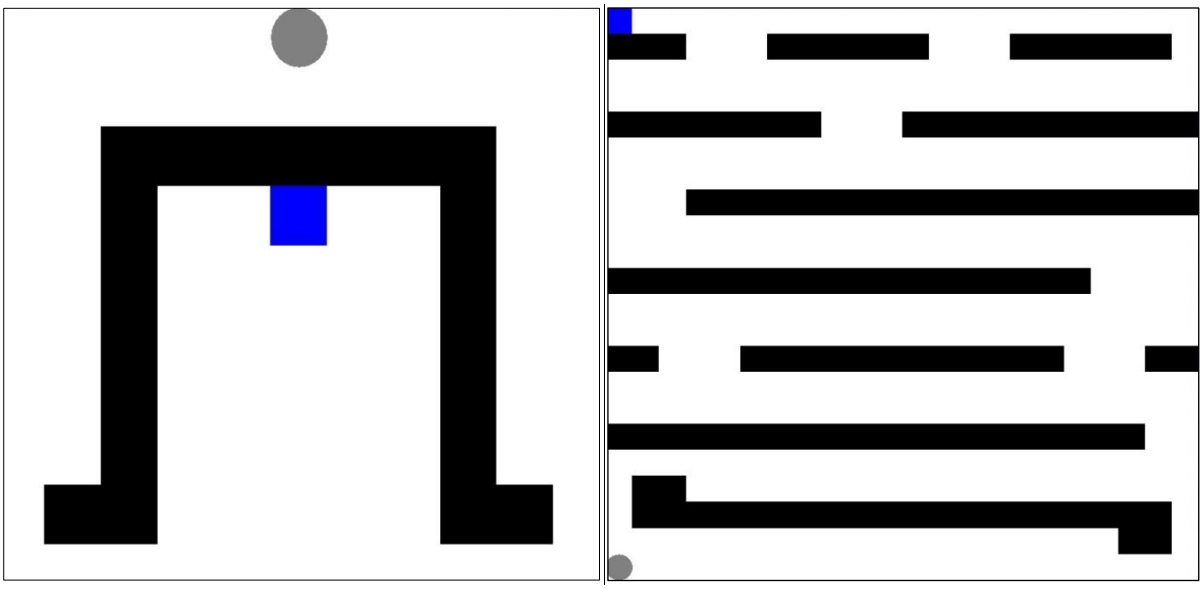

Figure 1: Grid Worlds. Left-to-Right: (a) Easy Grid World (11X11 states); (b) Hard Grid World (22x22 states)

Implementation Methodology: All the experiments described in this report were carried out using the Brown-UMBC Reinforcement Learning and Planning (**BURLAP**) java code library[1]. Also, the code used is based on a code base developed by Juan Jose[2] (an OMSCS student).

[1] http://burlap.cs.brown.edu/

Grid World Dynamics Model: For both grid worlds, the agent can move in one of four directions (north, south, east, or west), and its motion is stochastically determined. Specifically, the probability that an agent moves in the direction it intends to is 0.8, while there is a probability of 0.667 respectively that it moves in a direction other than its intended direction of motion. This is collectively known as the **state transition function**.

Also, both grid worlds are configured such that the reward received at the destination state is 100, while at every other state the reward received is -1. This is known as **the reward function**.

Easy Grid World - Experiments and Analysis: PI, VI and QL were each used to solve the easy grid world. To enable accurate comparison of the 3 sets of results, each RL technique was run using the same configuration (i.e. discount factor = 0.9), and run for 1000 iterations. For each technique the execution time per number of iterations, number of steps to reach the terminal state and the reward per number of iterations were recorded and analysed. Please note that in order to show the results clearly, only 500 iterations are plotted in the charts below. Also note that all the QLearners in this paper are configured to perform greedy exploitation, with no exploration (i.e. epsilon = 0).

Figure 2 below shows plots of steps and rewards per iteration for PI, VI ad QL. The first remarkable point is that the two plots are virtually horizontal mirror images of each other. This indicates that the fewer the steps taken by the agent, the higher the rewards achieved. Considering that for the easy grid world all states other than the terminal state have negative rewards, this is to be expected.

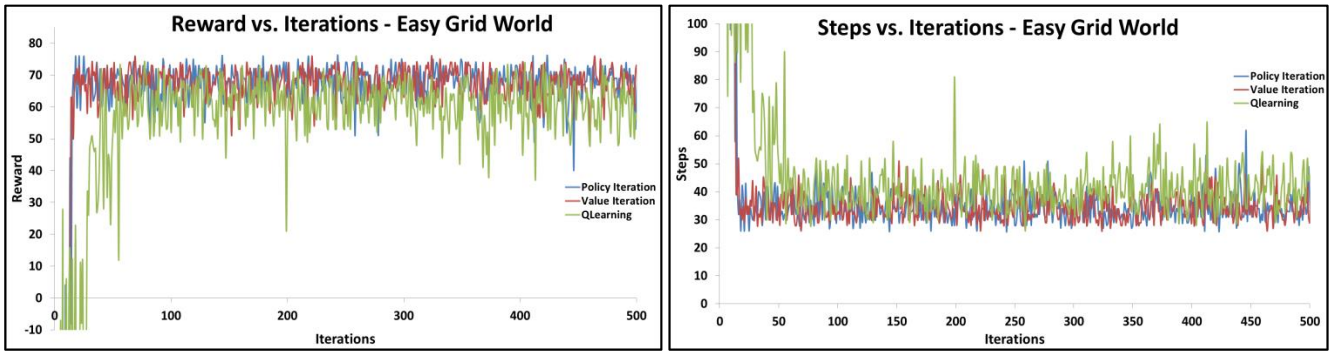

Figure 2: Easy Grid World Analysis. Left-to-Right: (a) Rewards vs Iterations; (b) Steps vs Iterations

Furthermore, **Figure 2** allows us to estimate when each of the techniques converges. Both VI and PI converge after **17 iterations** in both charts. Also, considering the stochastic nature of the experiment, VI and PI both appear to require around the same number of steps (**25 – 45 steps**) and both lead to approximately the same level of reward (**55 – 75 points**) after convergence. To reiterate, this wide range of value can be attributed to the stochastic nature of the agent's movements in both algorithms

In contrast, QL requires many more iterations before it begins to approach convergence (**40 iterations**). Also, after convergence QL still shows a wider range in the number of steps it requires **(30 – 52 steps**) and the level of reward achieved (**50 – 72 points**). Similar to VI and PI, the variance in the range of values here is due to the stochasticity of the agent's movements.

[2] https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4

Figure 3 below shows the policy calculation time per iteration for each of the techniques. The chart clearly shows that PI takes the most time to come up with a policy, followed by VI. QL on the other hand takes almost no time, regardless of the number of iterations run. Upon considering the manner of operation of the 3 techniques, this makes perfect sense. Value Iteration iterates over all the states in the grid world and updates them using the utilities of every reachable neighbouring state, until convergence is reached.

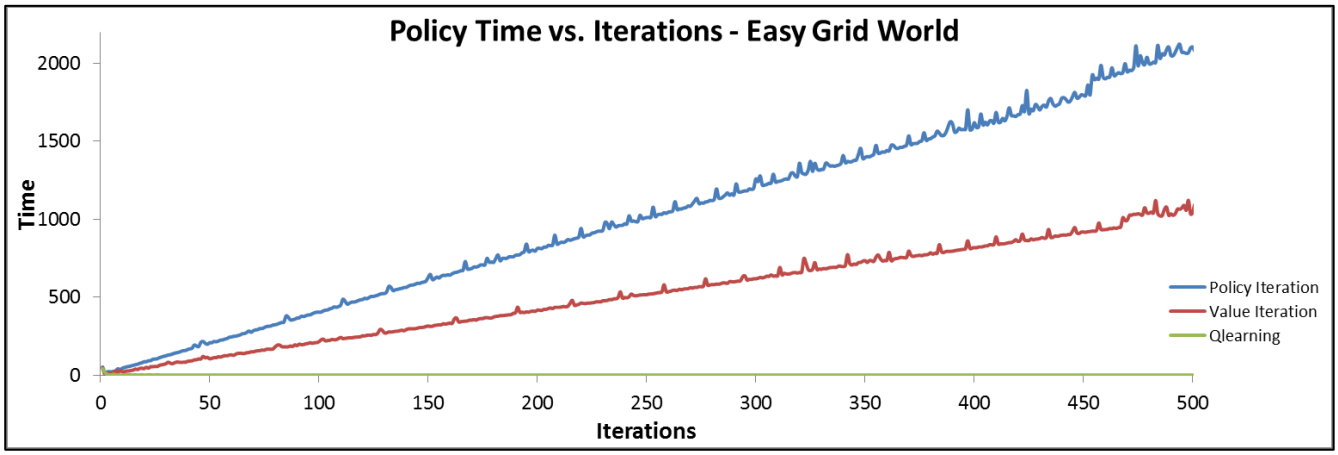

Figure 3: Policy Calculation Time vs. Number of Iterations – Easy Grid World

Policy Iteration, on the other hand, starts with a random policy for each state and calculates the utility of each state in the policy. In the next iteration, the policy is then updated to the policy that provided the maximum utility, using the utility values obtained in the last iteration. The utility calculation step in PI involves solving a set of linear equations, and the number of equation depends on the number of states in the grid world. This can be a time consuming activity, depending on the number of states involved. Therefore, while PI's modus operandi sounds very similar to that used by VI, VI calculates policies faster than PI because VI does not involve the time consuming utility calculation.

QLearning operates under a totally different paradigm, and does not require a model of the world it is trying to solve (i.e. no knowledge of the reward or transition function is required). Instead, QL simply keeps a record (a **Q table**) of the observed states, actions resulting states, and observed rewards, iteratively updates this table, and makes decision at every point based on the action that that provides the Q reward. Since all that is really being done by QLearning is updating the Q table, it runs extremely fast, regardless of the number of iterations used.

Finally, **Figure 4** below shows the resulting policies from the 3 techniques for the easy grid world. VI and PI arrived at the exact same policy. In other words VI and PI both converged to the optimal policy. On the other hand, QL produced a somewhat different final policy when compared to VI and PI. Examples of this can be seen in **Figure 4** by comparing the second and tenth columns of the QL policy with those from VI/PI.

This difference is probably due to the fact that QL did not converge to the optimal policy , and one expects that given enough iterations, QL would produce a policy that is more similar to those from VI and PI.

Hard Grid World - Experiments and Analysis: Similar to the easy grid world, PI, VI and QL were each used to solve the hard grid world, using the same configuration (i.e. discount factor = 0.9) for

each technique and run for 1000 iterations. As before, only a fraction of the iterations run are plotted below, to make the charts clearer and more visible (without loss of generality).

Figure 5 below shows the number of steps required to reach the terminal state, for different numbers of iterations. From the chart, we can see that the 3 techniques exhibit similar behaviour to that observed in the easy grid world (i.e. VI and PI converge much earlier than QL).

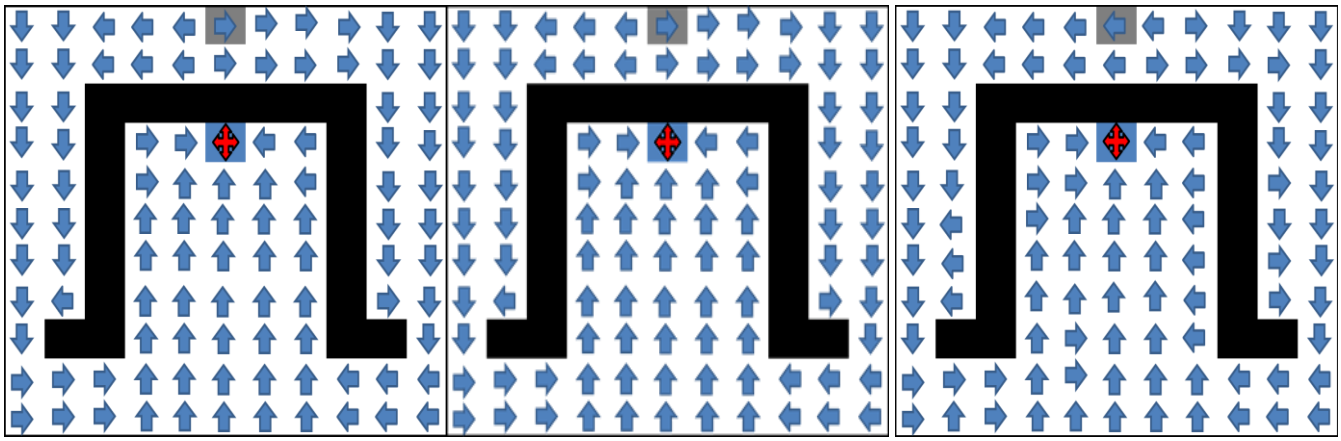

Figure 4: Easy Grid World Policies. Left-to-Right: (a) Value Iteration Policy; (b) Policy Iteration Policy; (b) QLearning Policy

There is a marked difference in the behaviour of VI and PI here though. Specifically, PI converges after **58 iterations**, while it takes VI a few more iterations to converge (**61 iterations**). On the other hand, QL takes more than twice as many iterations before converging (**153 iterations**). It is also worth pointing out that it takes many more iterations for VI, Pi and QL to converge in the hard world, when compared to the easy world (see **Figures 9** and **10** below). This also makes sense, as the hard world (with **484 states**) contains 4 times more states than the easy grid world (**121 states**).

Furthermore, after attaining convergence and factoring in the inherent stochasticity of the agents movement, both VI and PI require more or less the same number of steps to reach the terminal state (**90 – 122 steps**). The fact that the average number of steps is on average much greater than the number of steps required for the easy world can be explained using the same logic as earlier described (i.e. greater number of states in the hard world). QL also requires a much greater number of steps to reach the terminal state here (**97 – 182 steps**).

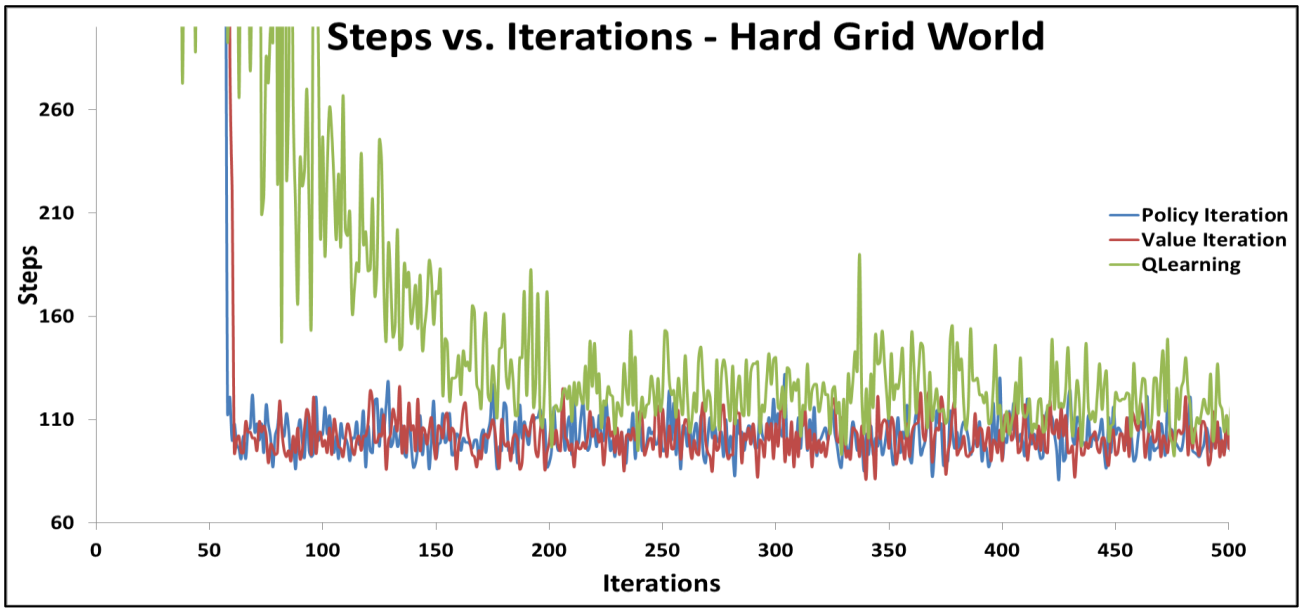

Figure 5: Hard Grid World Analysis - Steps vs Iterations

Figure 6 below provides further evidence of the number of iterations at which the different techniques converge. It also shows that, similar to the easy world scenario, VI and PI result in more or less the same reward after convergence (approximately between **-64 and 16 points**), while QLearning attains between **-44 points and 9 points**.

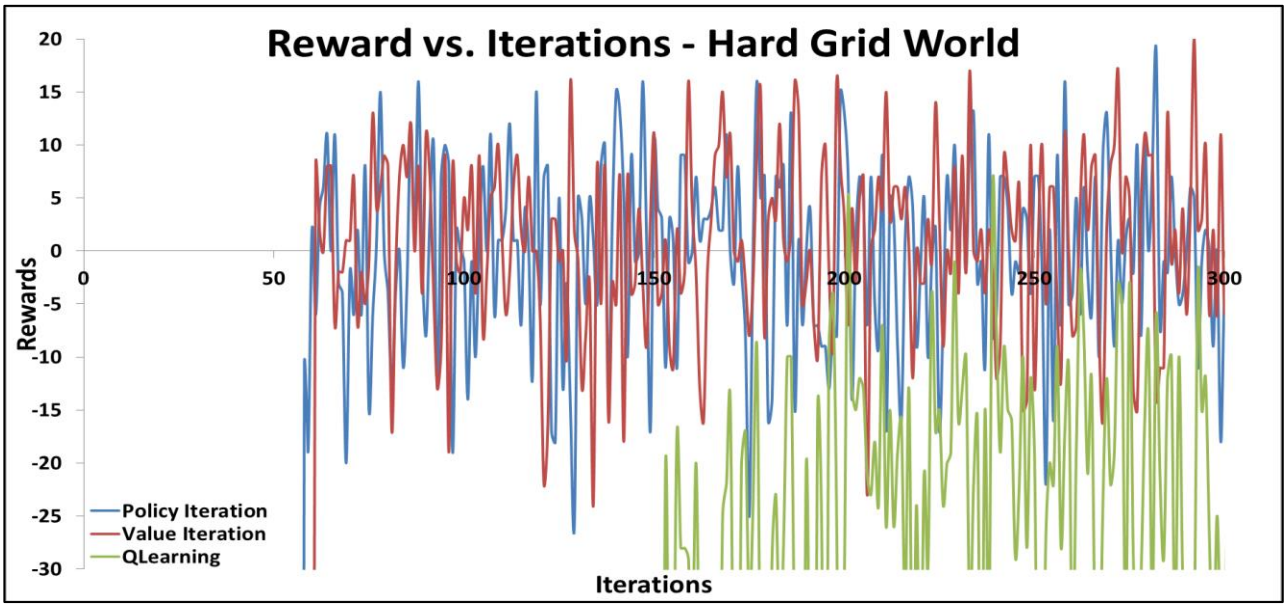

Figure 6: Hard Grid World Analysis - Reward vs Iterations

A couple of points are worth calling out here. First, the variation in the range of rewards achieved by all 3 techniques is much wider than in the easy world. I think this is because of the greater number of states in the hard world, which amplifies the stochasticity of the agent's motions. This in turn leads to a greater variation in the rewards achieved by all 3 techniques. Also, the QLearner seems to be much farther from convergence here than in the easy world. The explanation for this, in my opinion, is the same as the explanation given above.

Finally, it is worth mentioning that a chart of policy evaluation time vs. iterations for the 3 techniques (not shown) also revealed the exact same relationship as observed in the easy grid world scenario.

Figure 7 below shows the resulting policies from the VI and PI techniques for the hard grid world. A brief inspection of these policies show that both VI and PI produce exactly the same policy for the hard grid world, just like they did for the easy grid world (i.e. once again VI and PI converge to the optimal policy)

Figure 8 shows the output policy from QL for the hard grid world. This policy is very different from that produced by VI and PI. The policy also shows that the QL has not converged, as it abounds with examples of sub-optimal policy choices. Examples include the top right corner of the policy grid, where most of the states suggest that the agent should walk into a wall. It is also clear that the QL policy output here is much less optimal than that produced by QL for the easy grid world. In summary, QL once again fails to reach the optimal policy.

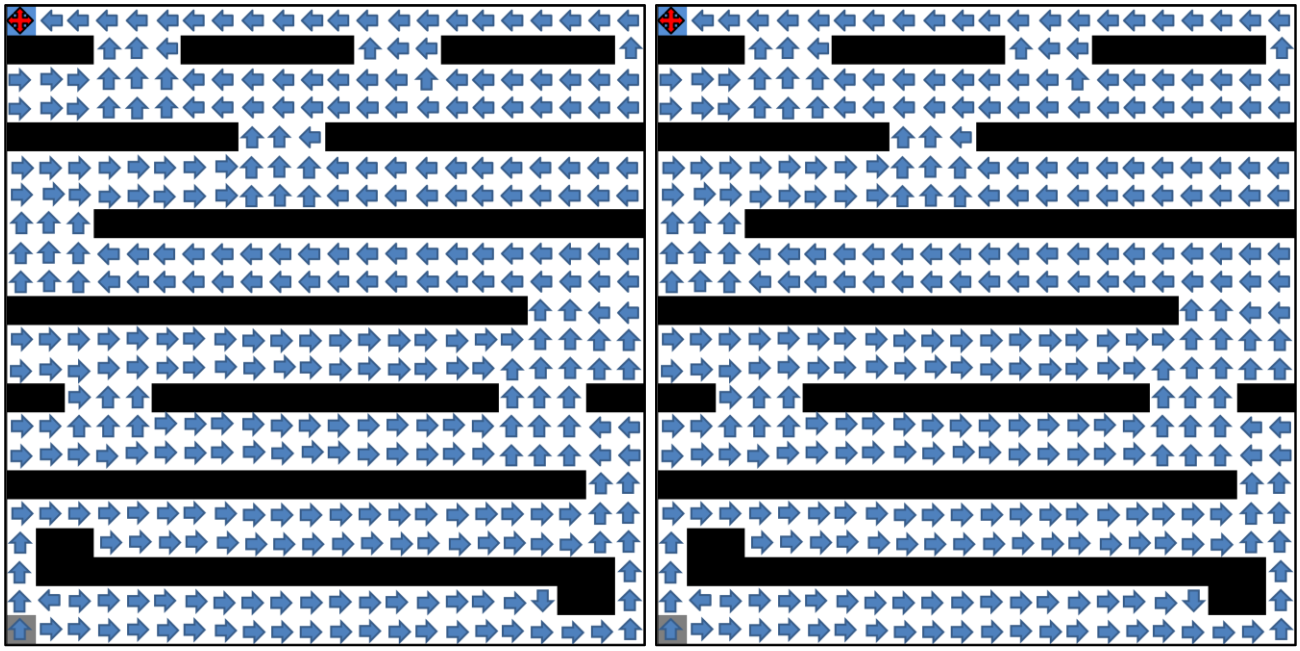

Figure 7: Hard Grid World Policies. Left-to-Right: (a) Value Iteration Policy; (b) Policy Iteration Policy

Investigating the Effect of Learning Rate on QLearning: The learning rate is a parameter that QL uses during its update cycle to determine how much weight to place on its current state, and how much to place on new information. The higher the learning rate, the greater the weight placed on new information, and vice versa.

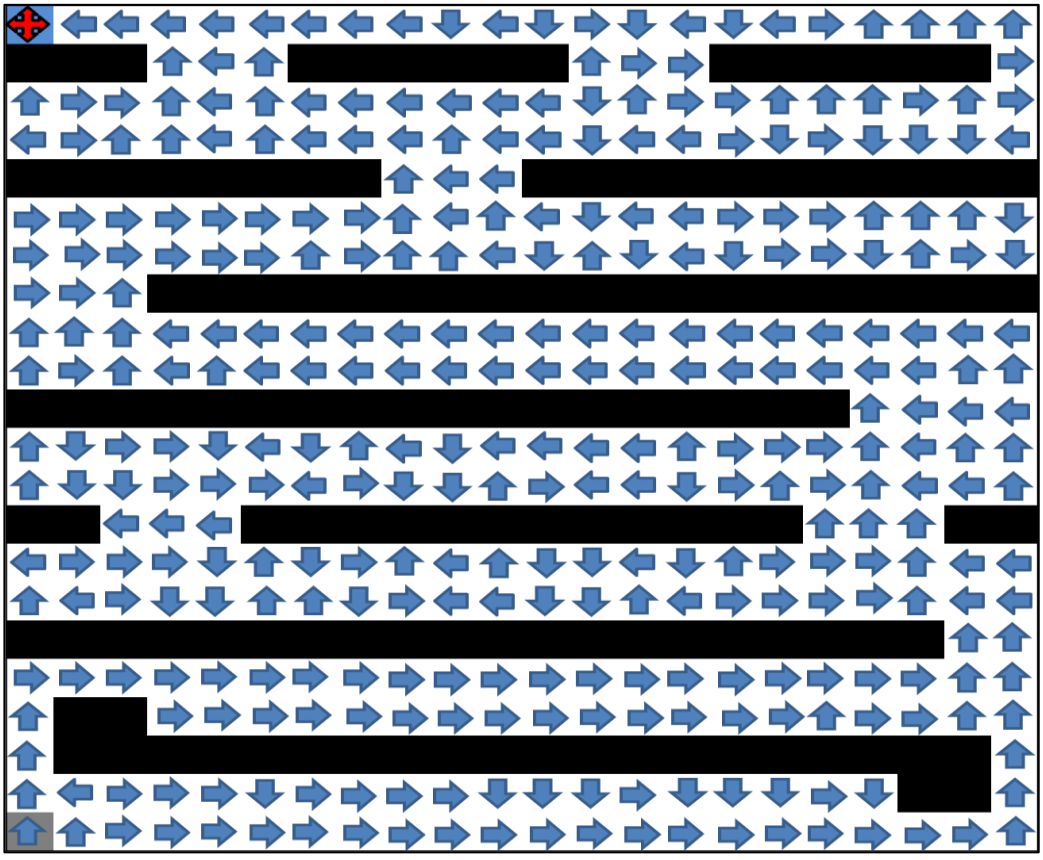

Figure 8: Hard Grid World - QLearning Policy

All the QLs in the foregoing experiments have been configured with a constant learning rate of **0.9.** In order to investigate the effect of learning rates on the operations of QLearners, I ran multiple QLearners on the hard world grid problem, with learning rates of **0.1, 0.3, 0.4, 0.7** and **0.9**, whilst keeping all other parameters constant (i.e. 1000 iterations and a discount rate of 0.9). **Figure 9** below shows the results of this experiment.

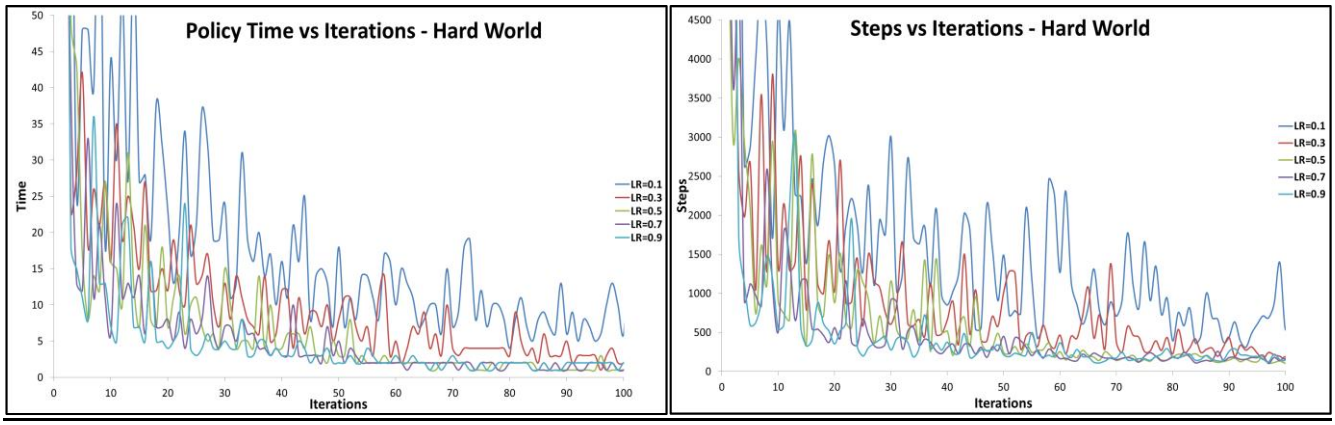

Figure 9: Effect of Learning Rates on QLearning. LTR: (a) Policy Calculation Time vs. Number of Iterations – Hard Grid World; (b) Steps to Terminal State vs. Number of Iterations – Hard Grid World

From **Figure 9a**, we can clearly see that as the learning rate increases, the agent requires less time to calculate and output a policy. Also, **Figure 9b** shows that as the learning rate is increased, the agent tends to reach convergence with fewer iterations, and also displays less variation on the number of steps required to reach the terminal state.

Both of these charts seem to suggest that a higher learning rate is more desirable. However, care must be taken when setting the learning rate, as setting it too high might lead to overly speedy convergence and results in the QLearner settling for a local maxima.

<u>Conclusions</u>: **Figures 10 and 11** below show a summary of the results VI, PI and QL on the easy and hard grid worlds. From **Figure 10a**, we can see that all 3 techniques require fewer iterations to converge in the easy grid and more iterations to reach convergence in the hard grid world. Furthermore, the chart shows that VI and PI require the same number of iterations to converge in the easy world. Conversely, PI requires fewer iterations to converge than VI in the hard grid world. Finally, **Figure 10a** shows that for both the easy and hard worlds, QL requires may more iterations to reach convergence than VI and PI.

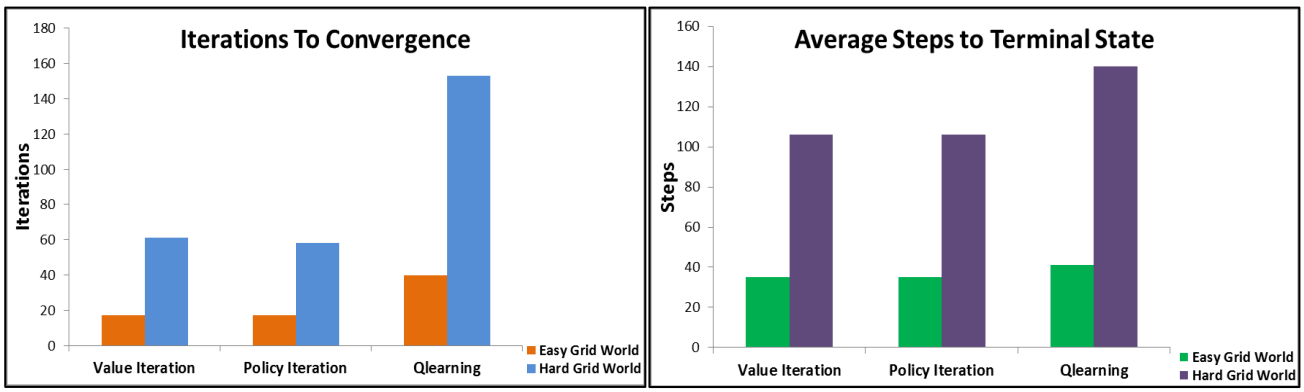

Figure 10: Summary of Performance of VI, PI and QL LTR: (a) Number of Iterations Required to Reach Convergence for Easy and Hard Grid Worlds; (b) Average Number of Steps Required to Reach Terminal State for Easy and Hard Grid Worlds

In the same vein, **Figure 10b** shows that on average VI, PI and QL take fewer steps to get to the terminal state in the easy grid world than they do in the hard grid world (due to the great difference in the size of the worlds). It also shows that both VI and PI require essentially the same average number of steps to reach the terminal state in both worlds.

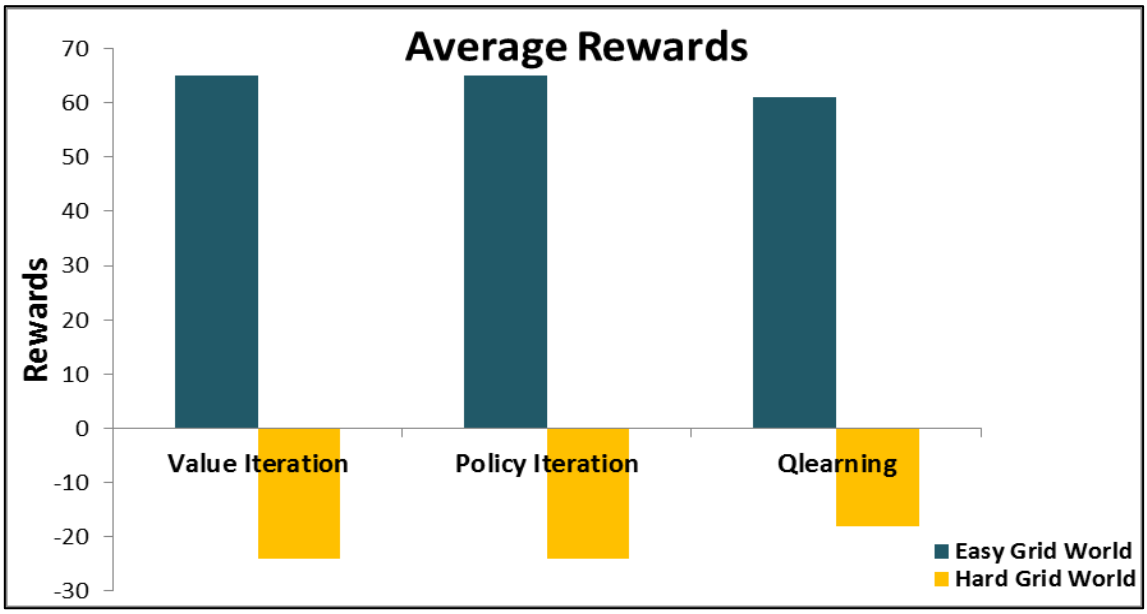

Figure 10: Average Rewards for VI, PI and QL on Easy and Hard Grid Worlds

Interestingly, **Figure 11** shows that on average, the easy grid world leads to greater rewards for all 3 techniques than the hard grid world. In fact, the average reward achieved by all 3 techniques in the hard grid world is negative. This is a peculiarity of the hard grid world I created, and occurs because all reachable states in the grid world have a reward on -1, coupled with the fact that there are over 400 states in the hard grid world.

Areas for Future Improvement: One of the major challenges I experienced was the limited memory I have on my computer. This prevented me from performing experiments with larger grids sizes, and also limited the number of iterations over which I could run VI Pi and QL. I believe that this limitation led to less than optimal results. For example, my research indicates that PI should typically require a smaller number of iterations to converge than VI. However, the difference between these 2 techniques is barely noticeable from my experiments.

As such, given the opportunity, I would like to perform these experiments on much larger grids (e.g. 100x100), and for much longer iterations (e.g. 10,000 to 100,000 iterations), and see if my observations are any different.

Another facet of QLearners that I did not get a change to explore due to time constraints is the exploration-exploitation dilemma. As earlier mentioned, all the QLearners in this paper were set to perform greedy exploitation (i.e. epsilon = 0). I would therefore welcome the opportunity to use different values of the epsilon parameter to stochastically determine what action my QLearners take, and see how this affects the behaviour of the QLearners.