Nirave Kadakia

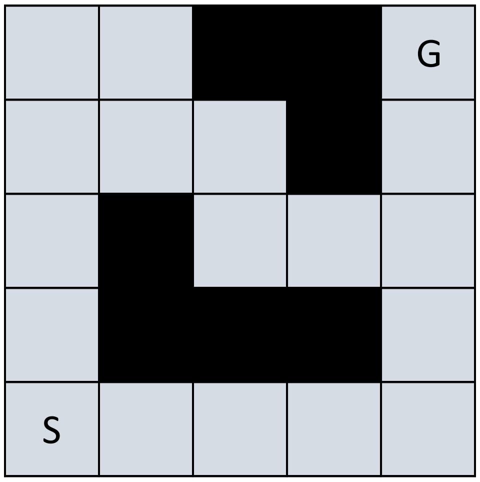
Exploring Markov Decision Processes

The purpose of this assignment is to compare and contrast two different Markov Decision Processes using Value Iteration, Policy Iteration, and a particular reinforcement algorithm, in this case, Q-Learning.

Types of problems:

Small Grid World

The first problem is classified as a small problem in that it has a small number of states. This problem is a simple maze based off a 5x5 Grid world. This maze is the following:



■	Wall
S	Start
G	Goal

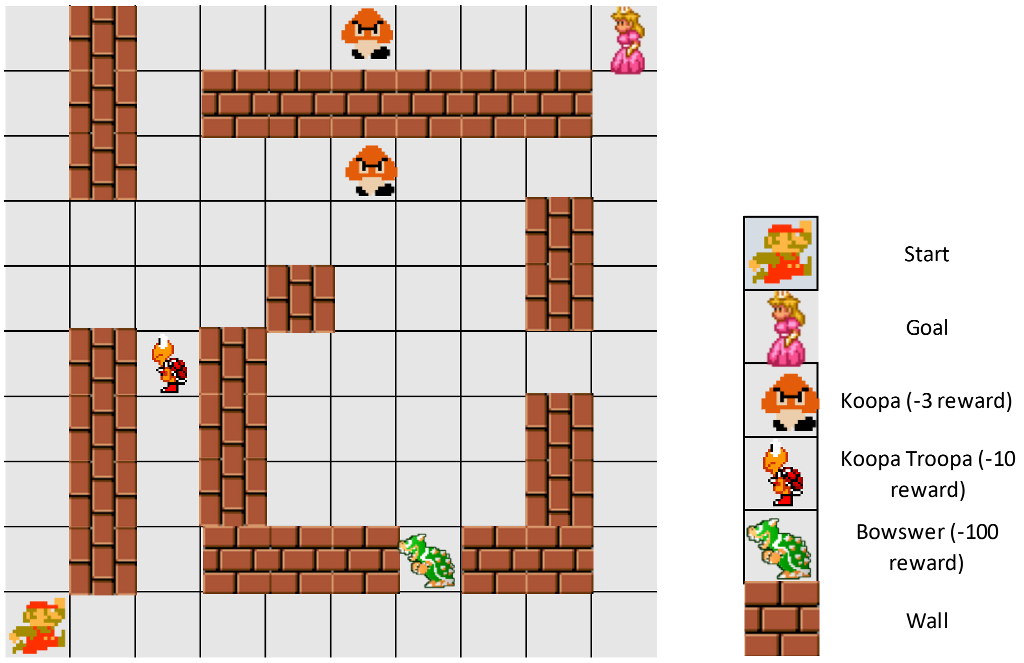
The maze is simple but stochastic, meaning that any intended direction, left, right, up, or down, the player will go in that direction 80% of the time. The rest of the time, it will choose, based on evenly split probability, any of the other directions. If the selected move results hitting a wall or moving off the grid world, the move will not be made.

The reward (goal) is 100 points, and each step is -1. So, it is quite easy to see the optimal rewards being 90 along the best path.

The reason why Grid world is important is that it can represent a lot of problems. Shortest path, driving a car, determining optimal routes, etc. Grid worlds are not just mazes, but can represent a plethora of interesting problems that reinforcement learning can handle. Solving for Grid worlds, in other words, solves a lot of problems in general.

Big Grid World (Super Mario Bros)

This is a larger grid world with many of the same properties as the small grid above. The reason why it was chosen was to ensure that an apples to apples comparison could be made to observe any differences between these two problems.



The purpose of this is to emulate the old Nintendo game, Super Mario Brothers, in a larger, 10x10 grid, where Mario, on the bottom left, has to find the princess on the top right. And Mario would like to avoid his enemies – Goombas (mushroom), Koopa Troopa (turtle), and Bowser (turtle with spikes) – thus there are larger negatives in those states. What is the best route to take, assuming the same stochastic methods above, to save the princess? The reward (goal) is 100 points, and each step is -1, the enemies above vary from -3 to -100. As shown below, and can be approximately confirmed visually, the optimal reward is 72.

Types of Techniques:

Value Iteration

Value iteration assumes that the true value of each state is known and finds the maximum utility via the following formula:

$$U(s) = R(s) + \gamma max_a \sum_{s'} T(s,a,s')U(s')$$

This value is calculated iteratively by doing the following:

- Assign a value to each state randomly
- For each state, calculate the utility based on the neighbors (using the formula above)
- Repeat these steps until values change less than a specific amount (which signifies convergence)

This should yield an optimal utility of each state, which the best policy can be taken from.

Policy Iteration

Policy iteration first creates a random policy for the world. Then it is an iterative policy that iterates by doing the following:

- Compute the utility of each state for the policy
- Computer the state utilities based on the formula above
- With new utilities, select the optimal action for each state
- Repeat until converged (i.e. no more/very little changes)

This returns a policy. This should, in theory, be quicker most times because it does not calculate all the utilities, but just finds a good policy.

Q-Learning

Q-Learning is a reinforcement algorithm that is model free, in that does not know the entire model beforehand. Instead, it relies on learning from knowing the current state, the actions, and the rewards for next state to determine the optimal policy.

It works by finding Q-values of a state. The basic algorithm states that at each step s, choose an action that maximizes Q(s, a), where Q(s, a) is the immediate reward for taking an action plus the best utility (Q) for the new state.

Implementation

The following code is based off Burlap, with Juan Jose's abstraction on top of Jython abstraction taken from Slack by eobrien31. See the references below for the source code links.

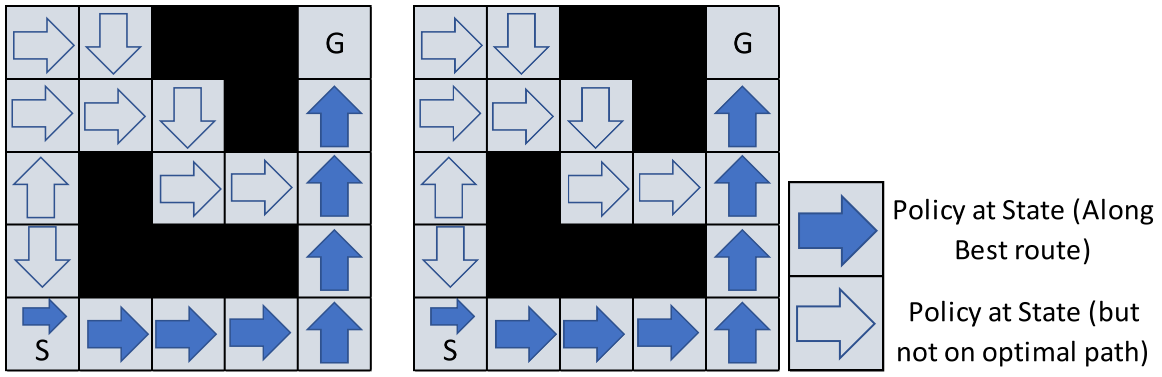
Small World Comparison

The small grid world is run for all three techniques.

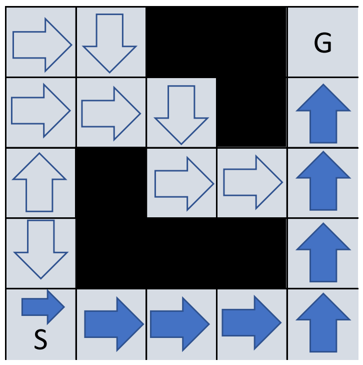
Optimal Policy Results

For Policy Iteration and Value Iteration, the optimal results return are the same.

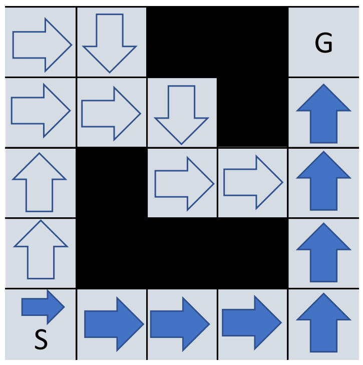
Note that Decay Rate was set to 0.99 as this was found to find good results.


G
S
G
S
Policy at State (Along Best route)
Policy at State (but not on optimal path)

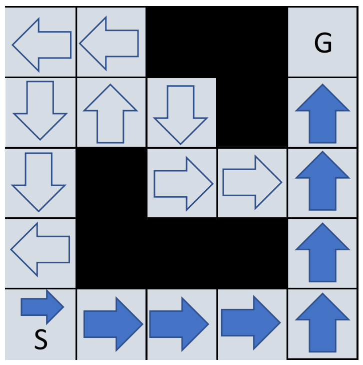
And the Q-Learning algorithm yields the optimal policy, but only with a certain Learning and Epsilon values:



Q-Learning L0.9 E0.5



Q-Learning L0.5 E0.5



Q-Learning L0.9 E0.1

With epsilon being high enough (i.e. non-greedy), Q-Learner can also find the optimal solution.

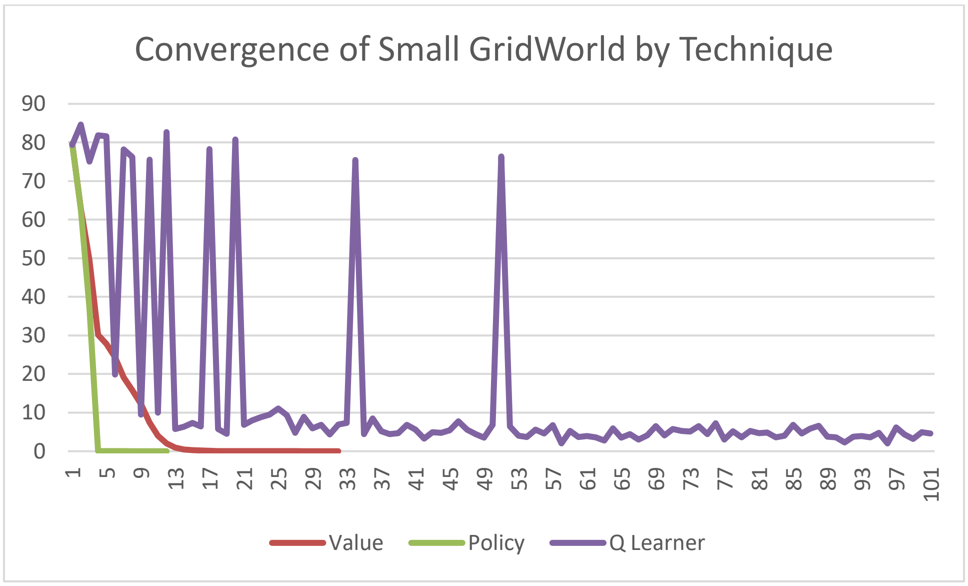
Convergence Results

Convergence for Value Iteration happens fairly rapidly for both. For Value Iteration, within 15 iterations, the convergence numbers (the change from one iteration to the other) becomes increasingly small, and the strictest definition of convergence (no change from the last iteration) happens at 34 iterations. Policy iteration happens in much fewer iterations, as expected. Convergence number decrease dramatically within 5 iterations and complete convergence happens after just 12 iterations.

As stated earlier, Policy Iterations will be much quicker, in terms of iterations, because an optimal policy is found – as opposed to Value Iteration, which computes policy indirectly by computing the optimal utility, then finding the policy that fits it.

Q-Learning takes more iterations. For example, with the Learning Rate at 0.9 and Epsilon at 0.5 it takes approximately 15 iterations to stabilize mainly because it is model free and must learn as it goes. Also note that Q-Learner never truly convergences within 100 iterations. In fact, the entire algorithm did not converge after 1000 iterations. However, stabilization occurs at around 60.

Below are the Convergence numbers of Policy Iteration, Value Iteration, and Q-Learner for the small grid for up to 100 iterations:

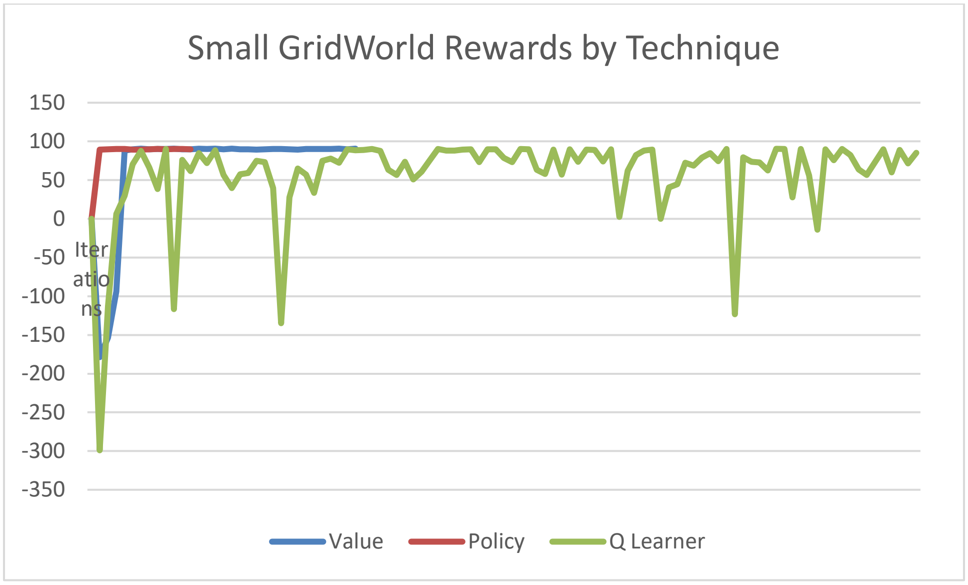


Total Rewards Results

Graphing Rewards provides a visual representation of when it convergences as well, signified by mapping any changes to the rewards, a.k.a policy costs below. The Reward (i.e. policy cost) max's is around 90, and all 3 techniques achieve it that mark, though Q-Learner tends to vacillate at or below it due to the random nature of Q-Learner.

We can see that all 3 techniques achieve quick results in a few iterations. Policy Iteration, as stated earlier, is the quickest, with Value Iteration also not so far behind. And with the limited number of states, Q-Learner learns quickly and achieves a good result somewhat early as well. This emphasizes the technique differentiation explained above.

Note the convergence, signified by the Policy Cost, or the total amount of rewards, for Value Iteration and Policy Iteration, reaches its maximum early on and stays around there and Q-Learner takes a little longer to reach and then is still not that stable around that value – thus verifying the convergence findings stated above.



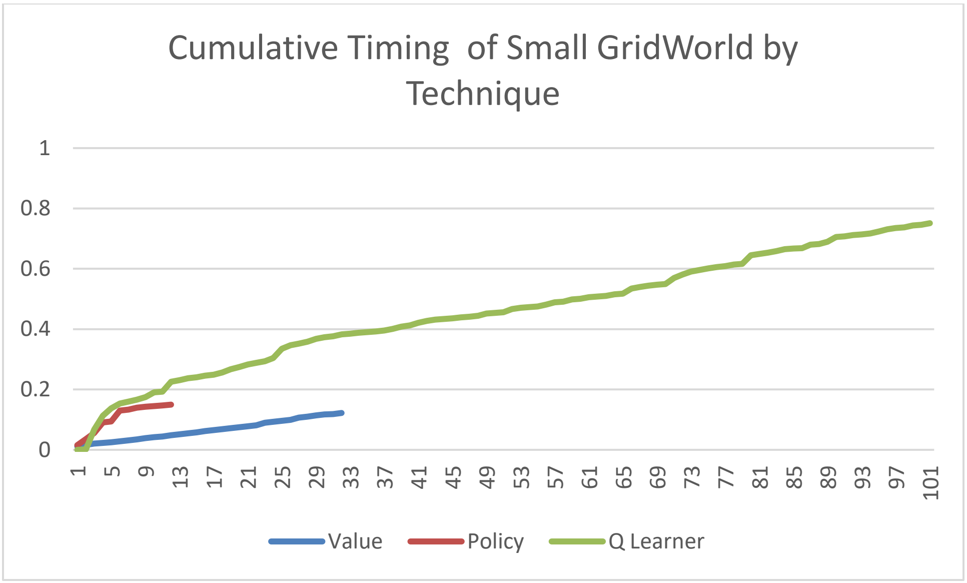
This makes sense as Policy and Value Iterations would take a small number iterations because both techniques know the entire world, and the number of states is small. Q-Learner, however, being model free, must take necessary steps in order to essentially learn the world, and as such, does take more time.

Timing Results

With such a small world, both Value Iteration and Policy Iteration are extremely quick, with Value being the quick with a total of 2.09 cumulative seconds to converge for those 33 iterations, and Policy taking less time at 1.27 seconds' total for 13 iterations. This makes sense because Policy iteration is quicker in that it does not calculate the utilities – so the result is nearly twice as fast to reach the optimal policy because of the reduced number of iterations.

Q-Learner, again, takes longer, as it is a model free state and naturally takes more iterations. It takes 12 seconds.

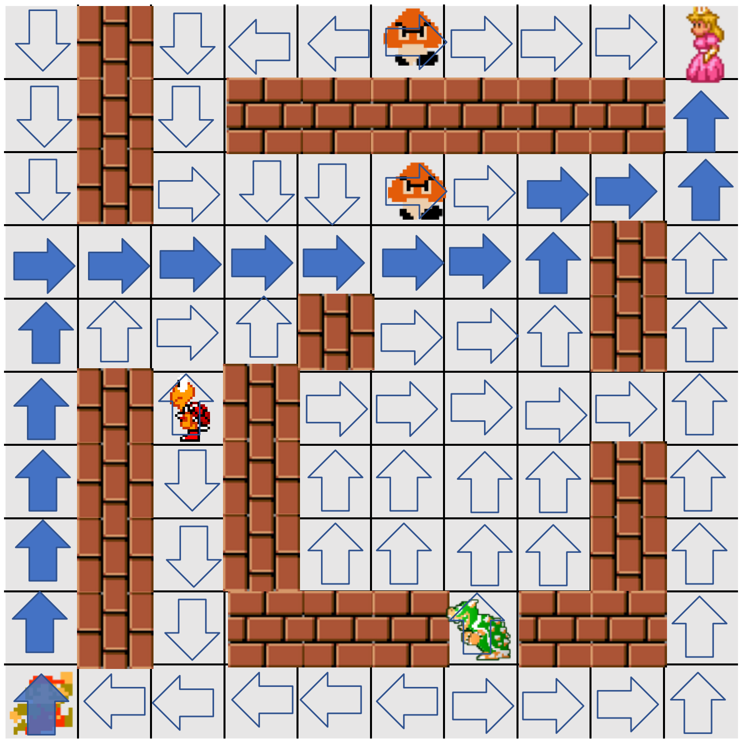
However, it is important to see time for each iteration. Policy iteration has a larger ramp-up time than Value Iteration or Q-Learning, but then amount of computation time needed for Policy iteration lowers dramatically near convergence. This may be attributable to the fact that Policy Iteration has to create policies, solve linear systems, and any improvement early on requires large changes to policy. Value Iteration and Q-Learning, on the other hand, appear to be running at constant time.

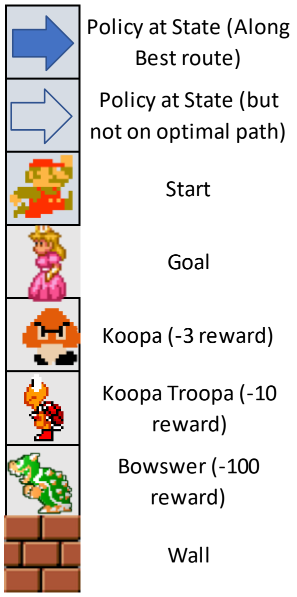


Big World Comparison

Optimal Policy Results

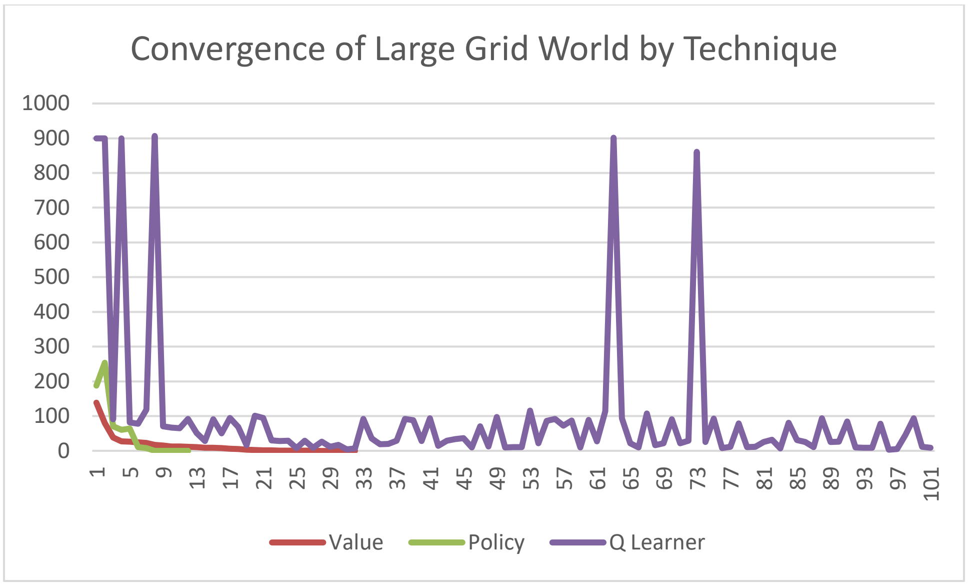
Note that all policies return the same optimal policy:




Policy at State (Along Best route)
Policy at State (but not on optimal path)
Start
Goal
Koopa (-3 reward)
Koopa Troopa (-10 reward)
Bowswer (-100 reward)
Wall

Convergence Results

Below are the Convergence numbers of Policy Iteration, Value Iteration, and Q-Learner for the large grid for up to 100 iterations:

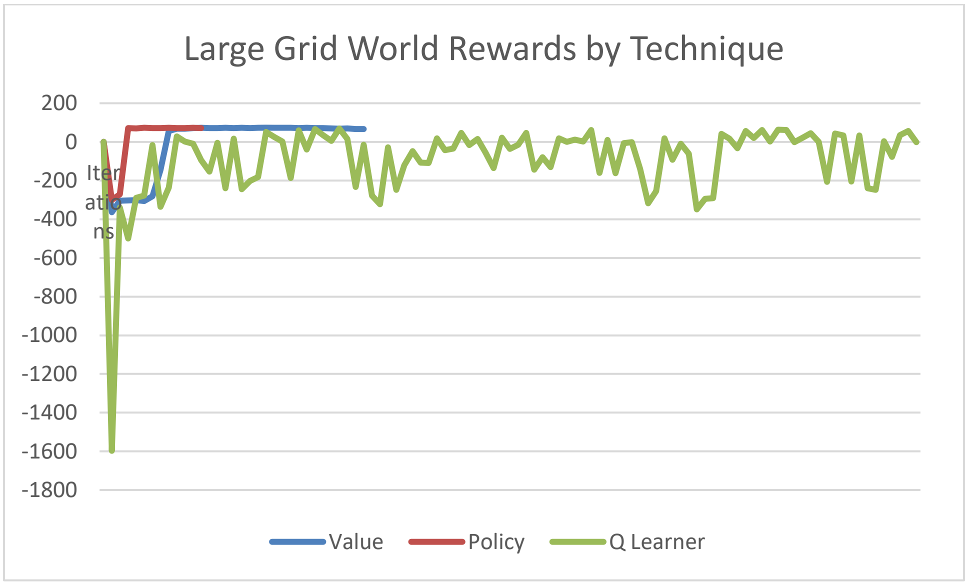


Observing the Large Grid World's convergence numbers verifies the same results as seen in the smaller Grid world.

Policy Iteration takes the quickest, followed closely by Value Iteration – as both take advantage of the knowledge of the world while Q-Learner takes longer as it is model free. And as stated earlier, Policy Iteration is quicker because it does not calculate values, just the policies.

Rewards Results

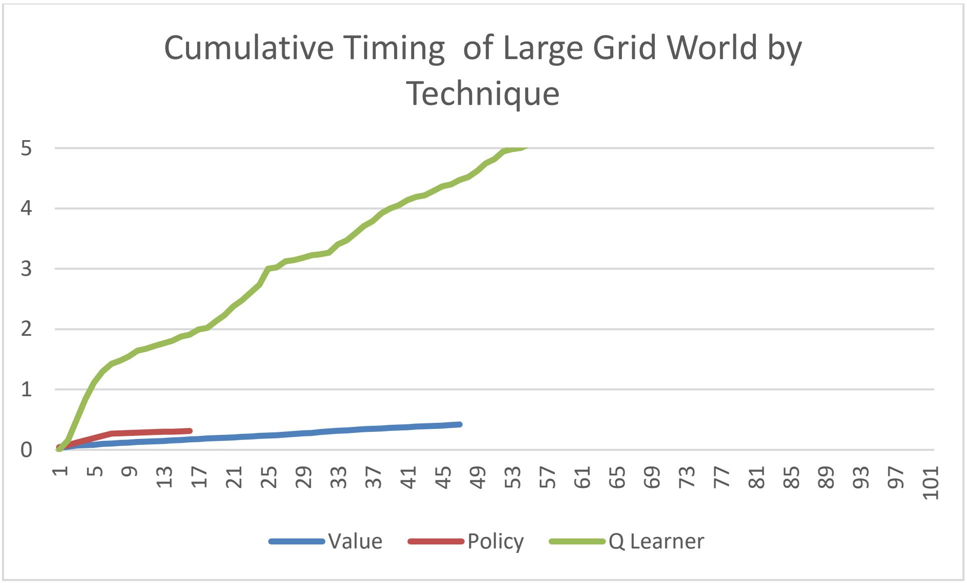
Below are the Total Rewards numbers of Policy Iteration, Value Iteration, and Q-Learner for the large grid for up to 100 iterations:



To reiterate the results, the values of the rewards mirror the results found in the smaller world. The increased state, however, emphasizes the gap in iterations needed to reach maximal rewards for Q-Learner. This makes sense that the gap would increase as the number of states Q-Learner has to learn about the environment has increased 4-fold.

Timing Results

Below are the Total Rewards numbers of Policy Iteration, Value Iteration, and Q-Learner for the large grid for up to 100 iterations:

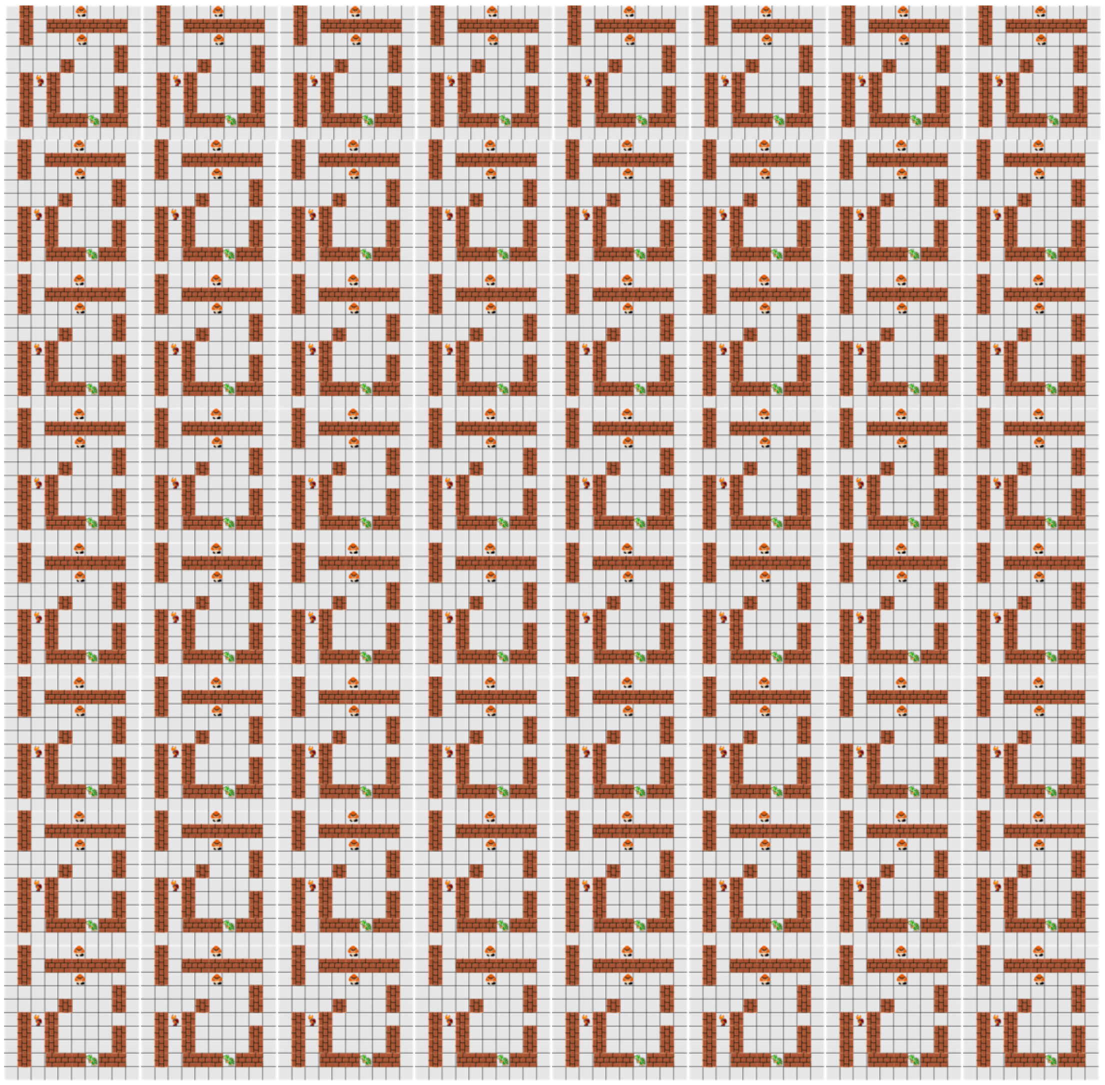


This somewhat validates the results from the small grid world, that Value Iteration and Q-Learner run in linear time. It also shows, slightly, that Policy Iteration has a larger ramp-up time.

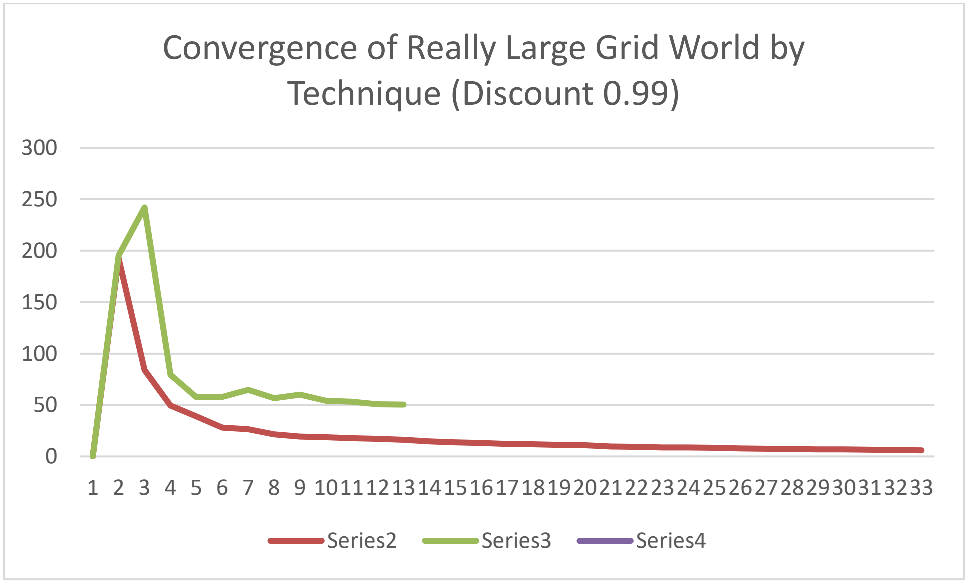
However, it is important to see the increase in time, through number of states for Q-Learner is not linear in terms of steps. According to Koening and Simmons, the worst-case complexity of Q-Learner is O(n^2) for Grid Worlds, and this confirms it. As the number of states grew from 25 to 100, the time for approximately 50 iterations grew from approximately 0.45 to 4.8 seconds, an approximate 10.6 increase in time – in line with exponential growth of around 16x.

Policy vs Value Large Size

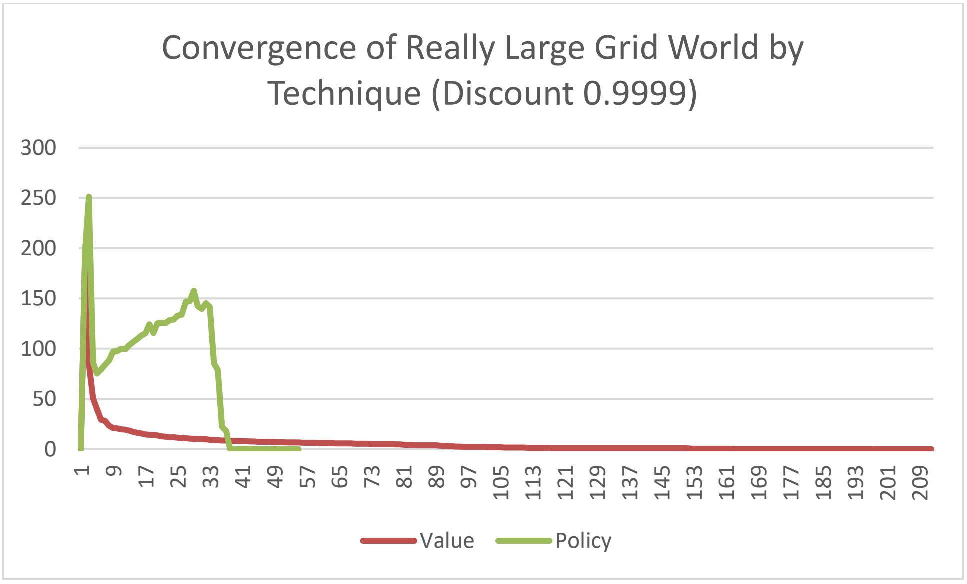
For a large world, Policy Iteration is not always quicker than Value Iteration. Unfortunately, due to computing constraints, the large size grid is not quite quick enough for Q-Learning. So, for the purposes of this paper, an 80x80 grid was created to compare Policy and Value Iteration only to see if there was a way to produce a grid where Policy Iteration was worse, in terms of iterations and time, than Value Iteration. The thought process was to see if an even larger world can create a reversal of the results we saw above. 64 of the large grid worlds were merged into an 80x80 Grid World to see if this was possible:



With a discount rate of 0.99, even with such a large size, the advantage of Policy Iteration vs Value Iteration was only magnified. Below shows the convergence difference at a discount rate of 0.99.



To ensure that the discount rate was not a factor, a different discount rate was tried. This was tried with a discount rate of 0.9999, as the worst case scenario for policy grows polynomially in 1/(1 – discount rate). Below is the graph:



This again produces results where the Policy may not necessary be the best choice if delta, to signify convergence, has to be value the of 0. Value Iteration converges after 212 iterations, whereas Policy Iteration convergence after only 54 iterations.

However, with the rapid decline in delta for Value Iteration, while Policy Iteration's delta remains high after 30 iterations, means that in this case, if the acceptable delta for convergence is high, Value Iteration does perform better than Policy Iteration. This reason why is that Policy Iteration solves a large system of Linear Equations for large state spaces. At 80x80, that's 1600 states, a large state space – thus we see Policy Iteration starting to falter compared to Value Iteration.

Conclusion

Both problems worked reasonably well with Value Iteration. However, as expected, Policy iteration works even quicker with the exception of extremely large grids. Both techniques also give the optimal policy. Q-Learning is a great tool that can also give the optimal policy, but the fact that it is model free means that it takes longer to find the answer, and convergence is not readily available.

References:

http://uhaweb.hartford.edu/compsci/ccli/projects/QLearning.pdf
https://www-s.acm.illinois.edu/sigart/docs/QLearning.pdf
http://idm-lab.org/bib/abstracts/papers/aaai93.pdf
http://www.cs.utah.edu/~piyush/teaching/29-11-print.pdf
Burlap: http://burlap.cs.brown.edu/
JuanJoses's github: https://github.com/juanjose49/omscs-cs7641-machine-learning-assignment-4
Eric O' brien's easymode jython code: https://files.slack.com/files-pri/T08LHBDJT-F52QLDZGT/download/assignment4clicktorun.zip