Project 4: Markov Decision Processes

Tyler Bobik
Georgia Institute of Technology

MDP problem set up

For this Project we are to come up with two interesting MDP Problems.

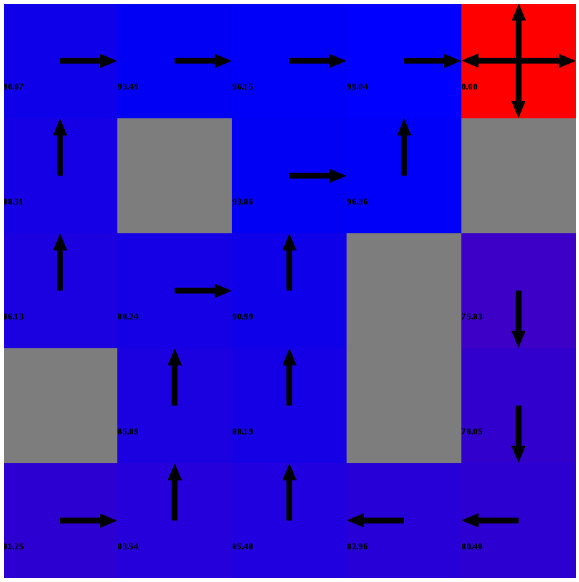
For the first problem with a small amount of states, it is only a (5x5) grid with 5 “walls” which are the grey boxes you see in the value iteration map below. For the first problem with a small number of states, I thought I came up with an interesting one, because there are technically two “optimal” paths that the agent can make it to the reward, both paths require the same amount of steps.

The second problem with a large amount of states, (11x11) grid. I thought this problem was interesting because as you can see there is a policy that is shorter on the, I was curious how the different algorithms will react to this. In tandem these problems are interesting because I will hopefully come up with results on how the algorithms preform on problems with small states vs. problems with large ones.

For both of these problems I will be using a probability of 0.8 that the agent will move in the correct direction and 0.2 probability it will move in a different direction. The agent additionally returns -1 for every state except if it is the goal state, then it returns 100.

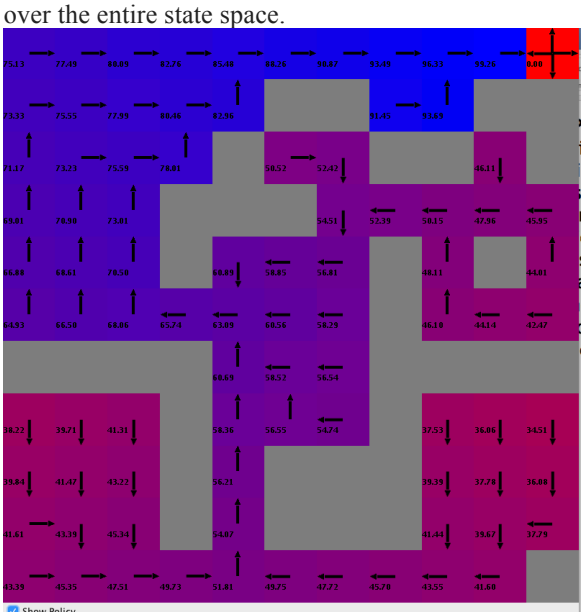
1.1 Value Iteration (Small # of States)

These values at each state corresponds to what the optimal future cost/reward trade off that you can achieve if we act optimally at any given state. I ran this test with max_iterations=100 and num_intervals=100. A "pass" in value iteration is VI running the Bellman operator over the entire state space.



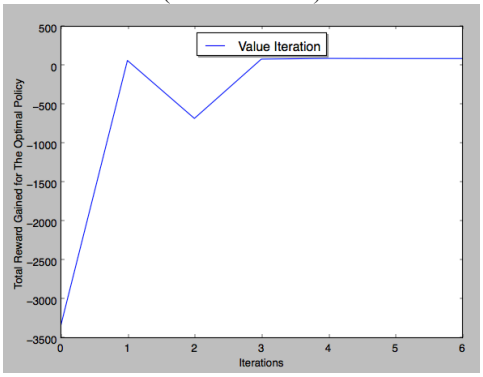
2.1 Value Iteration (Large # of States)

These values at each state corresponds to what the optimal future cost/reward trade off that you can achieve if we act optimally at any given state. I ran this test with max_iterations=100 and num_intervals=100. A "pass" in value iteration is VI running the Bellman operator over the entire state space.

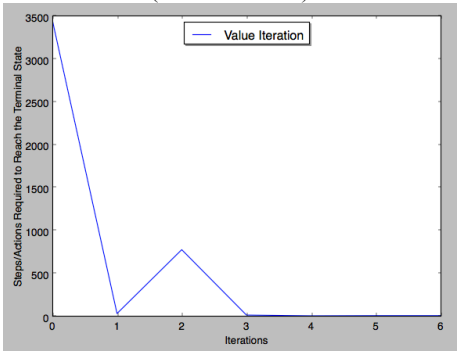


It should be noted that the numbers used for following graphs are averaged over 10 runs.

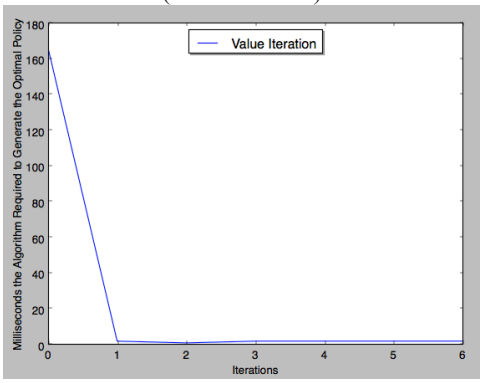
Steps/Actions Required to Reach the Terminal State
(1-7 Iterations)



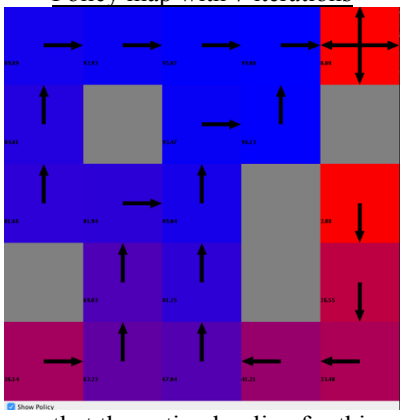
Total Reward Gained for The Optimal Policy
(1-7 Iterations)



Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-7 Iterations)



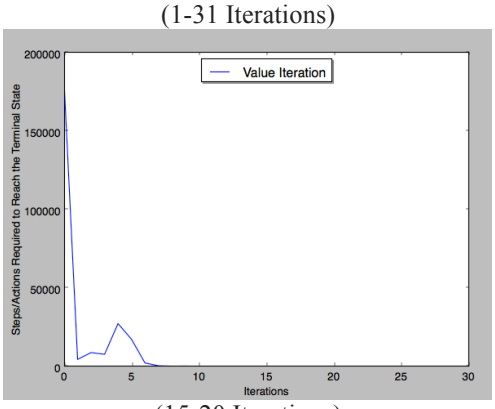
Policy map with 7 iterations



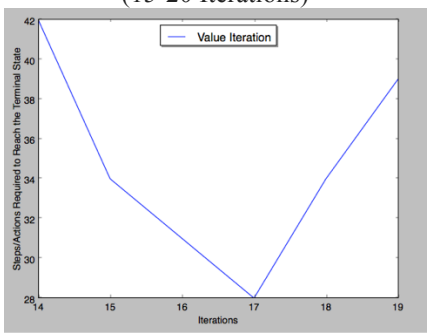
Here you can see that the optimal policy for this problem converges in 4 iterations (the graphs start at 0 iterations).

It should be noted that the numbers used for following graphs are averaged over 10 runs. I also had to zoom in to a specific range so you could see the convergence of the optimal policy this algorithm came up with.

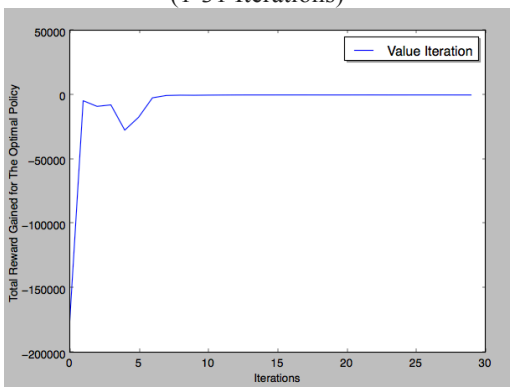
Steps/Actions Required to Reach the Terminal State
(1-31 Iterations)



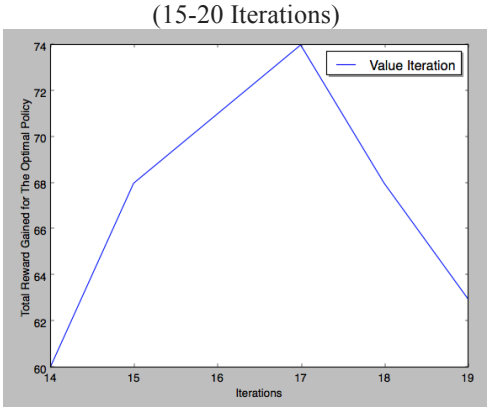
(15-20 Iterations)



Total Reward Gained for The Optimal Policy
(1-31 Iterations)



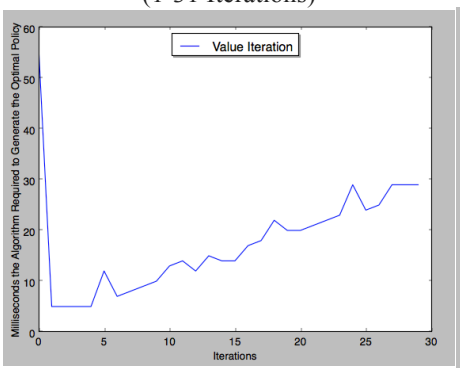
(15-20 Iterations)



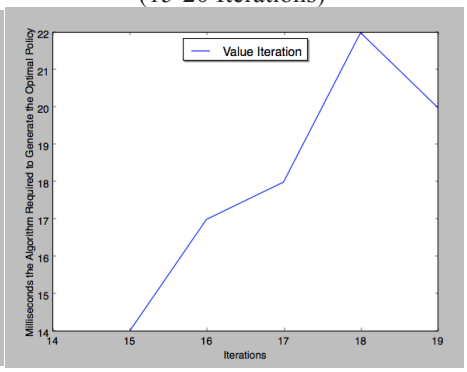
2.1 Value Iteration (Large # of States) (Continued)

Milliseconds the Algorithm Required to Generate the Optimal Policy

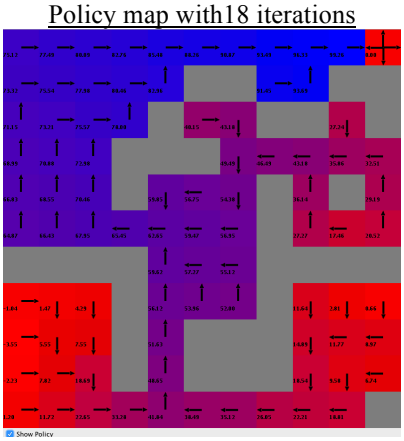
(1-31 Iterations)



(15-20 Iterations)



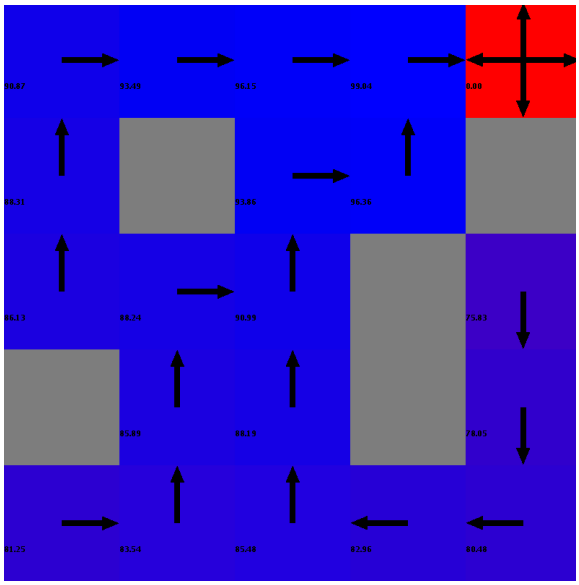
Policy map with18 iterations



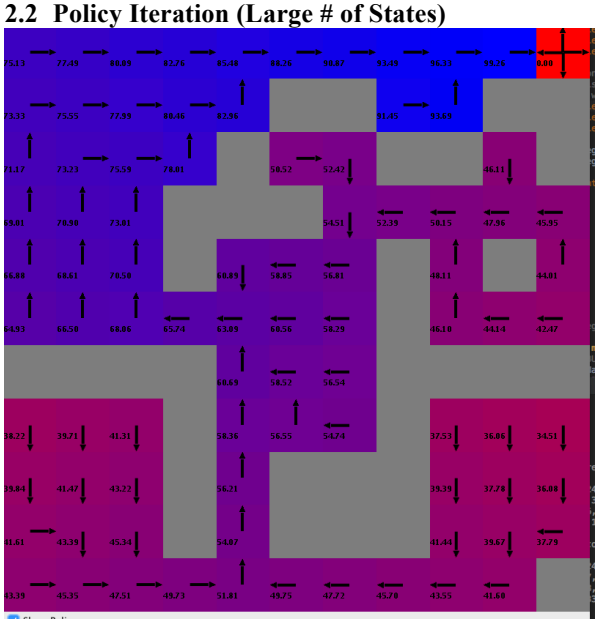
Here you can see that the optimal policy for this problem converges in 18 iterations (the graph starts at 0 iterations).

1.2 Policy Iteration (Small # of States)

A policy evaluation effectively runs value iteration to termination (but uses the fix policy Bellman operator since it's evaluating a policy), so each policy iteration runs value iteration a number of times. I ran this test with max_iterations=100 and num_intervals=100.

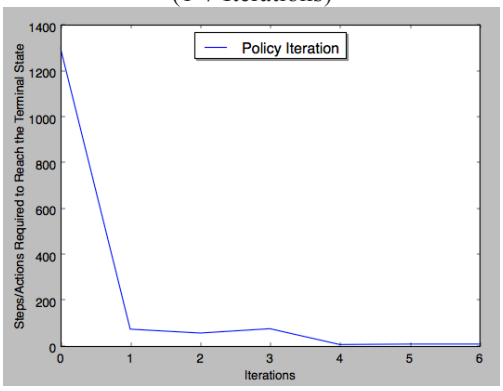


2.2 Policy Iteration (Large # of States)

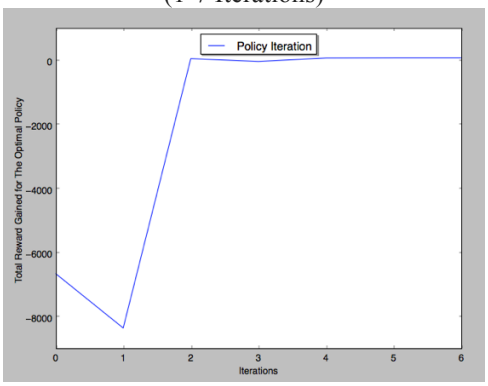


Here I showed one zoomed out graph and one zoomed in for each metric, so it is clearer to see what is happening.

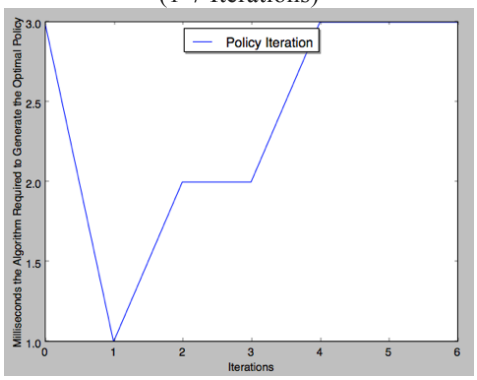
Steps/Actions Required to Reach the Terminal State
(1-7 Iterations)



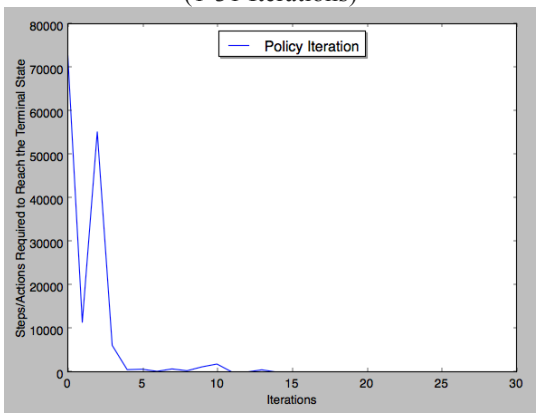
Total Reward Gained for The Optimal Policy
(1-7 Iterations)



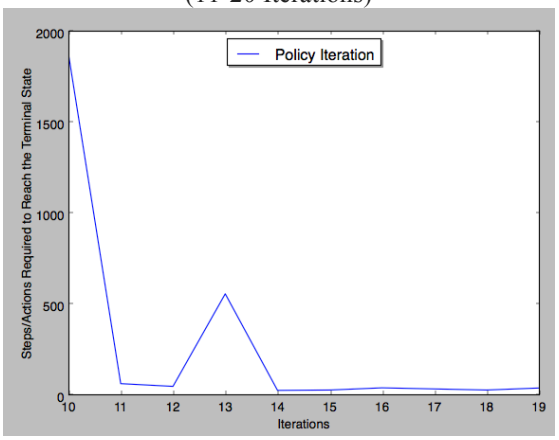
Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-7 Iterations)



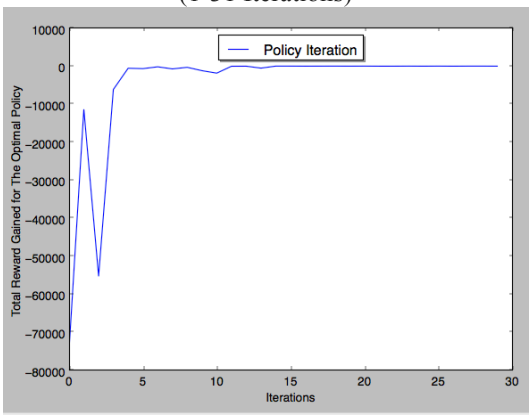
Steps/Actions Required to Reach the Terminal State
(1-31 Iterations)



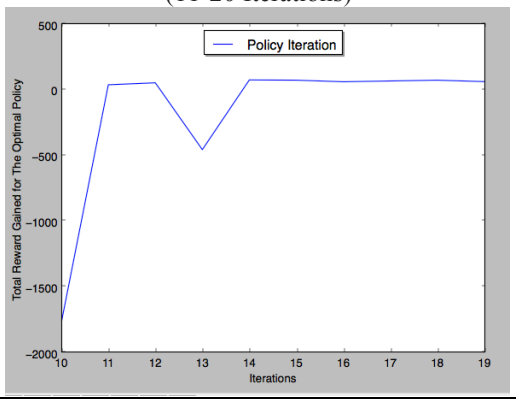
(11-20 Iterations)



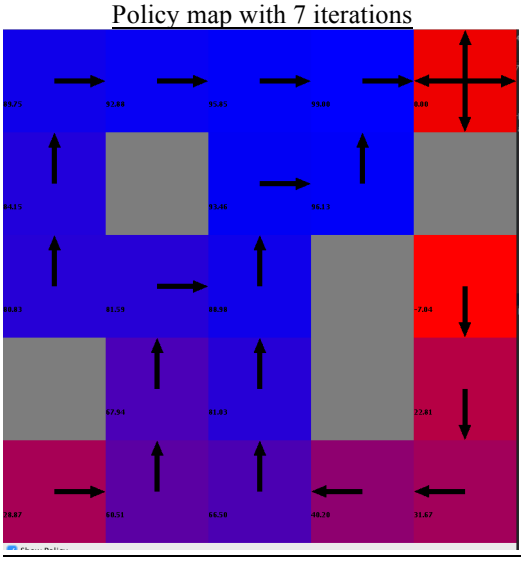
Total Reward Gained for The Optimal Policy
(1-31 Iterations)



(11-20 Iterations)



Policy map with 7 iterations

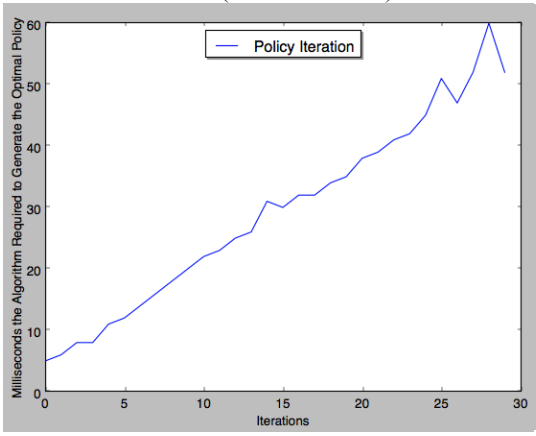


Here you can see that this algorithm converges on the optimal policy for this problem in 6 iterations (the graphs start at 0 iterations).

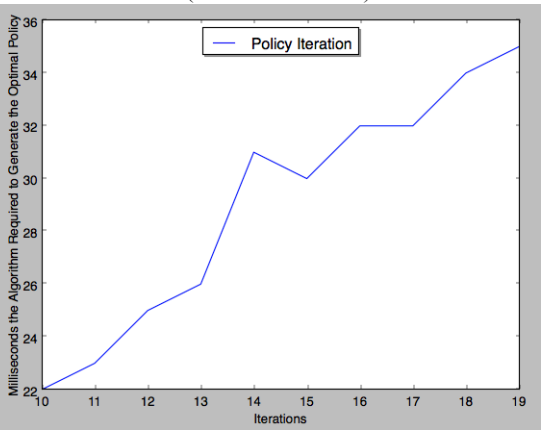
2. 2 Policy Iteration (Large # of States) (Continued)

Milliseconds the Algorithm Required to Generate the Optimal Policy

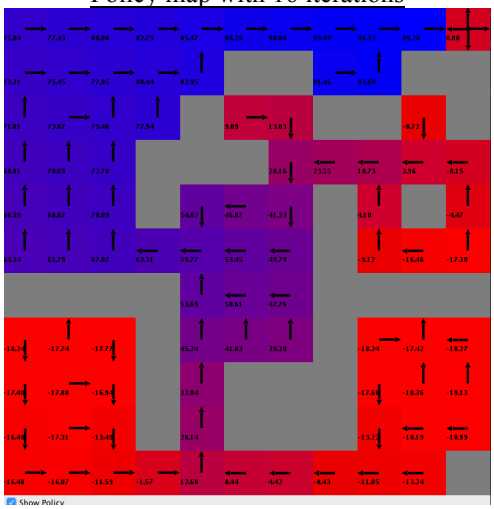
(1-31 Iterations)



(11-20 Iterations)



Policy map with 16 iterations



Here you can see that this algorithm converges on the optimal policy for this problem in 16 iterations (the graphs start at 0 iterations).

1.3 **Comparing the Two Algorithms**

Algorithm	Iterations Till Convergence	Steps/Actions	Time (Milliseconds)
Value Iteration	3	9	2
Policy Iteration	6	9	3

Here it is interesting that policy iteration did not out preform value iteration. Although they both ended up converging policy iteration took 3 more iterations and 1 millisecond more. I believe that value iteration did better because this problem space was small, at each iteration of evaluation of all states of your environment, you increment the value of a state depending on values of neighbor states, you do this until all your environment is covered. While with policy iteration, you evaluate the action of your process at each iteration, so that you improve your control law or policy. This takes more time to compute.

2.3 **Comparing the Two Algorithms**

Algorithm	Iterations Till Convergence	Steps/Actions	Time (Milliseconds)
Value Iteration	18	29	22
Policy Iteration	16	27	30

Here it is interesting that policy iteration outperformed value iteration in terms of the number of steps taken and iterations till convergence were both lower than value iteration, but the tradeoff is that policy iteration takes more time to perform.

Comparing Easy Grid World vs. Hard Grid World

It is interesting to see that for both easy grid world and hard grid world that policy iteration behaved very similarly to value iteration. In both grid worlds, the number of steps/actions required to reach the terminal state started very high, then improved with every iteration until they converge. In the easy grid world problem, value iteration does better because it the problem is simpler and value iteration can converge to the correct values faster because it finds the policy indirectly via the optimal value function. While on the other hand policy iteration finds a lower number of optimal steps in the harder grid world problem, this is because policy iteration manipulates the policy directly and starts with an arbitrary policy, improving the policy at each state, the tradeoff of using this algorithm is that it takes more time and this is apparent in both of the grid world problems where policy iteration takes more time to converge than value iteration.

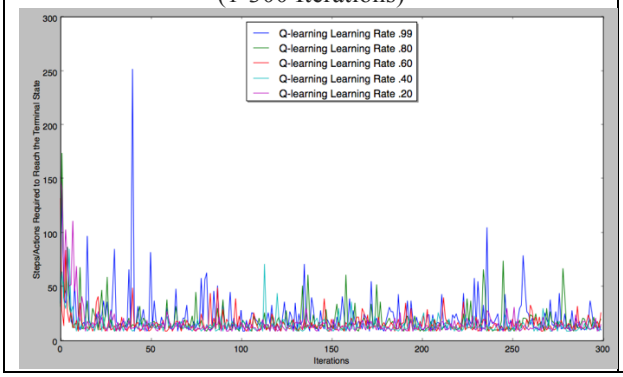
It is clear that the total number of possible states makes a difference in computation time and number of iterations till convergence.

Q-Learning

Learning rate used in the update rule of .9 and a gamma which is the discount rate used in the update rule of .9 and a qint which is the initial q-value to user every there of .99.
The learner always receives a reward of -1.0 until it reaches the goal, when it receives a reward of +1.0.

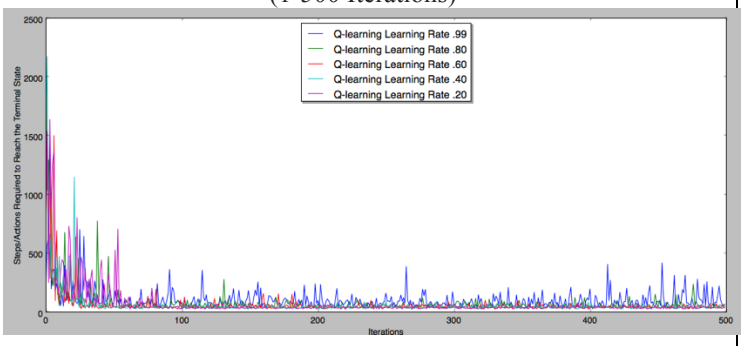
1.3 Q-Learning (Small # of States)

Steps/Actions Required to Reach the Terminal State
(1-300 Iterations)

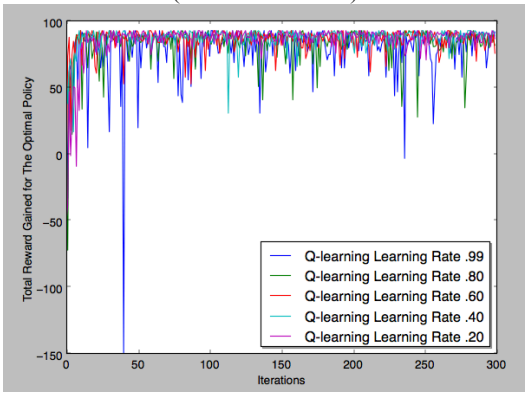


2.3 Q-Learning (Large # of States)

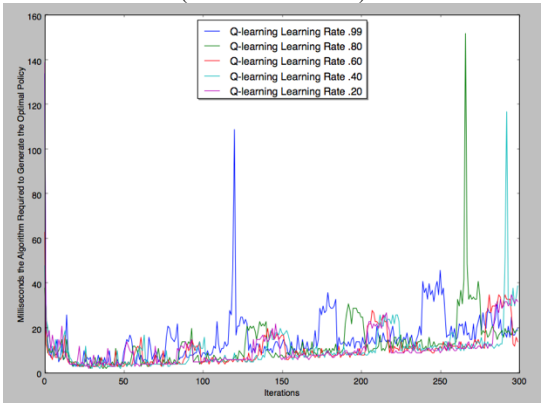
Steps/Actions Required to Reach the Terminal State
(1-500 Iterations)



Total Reward Gained for The Optimal Policy
(1-300 Iterations)

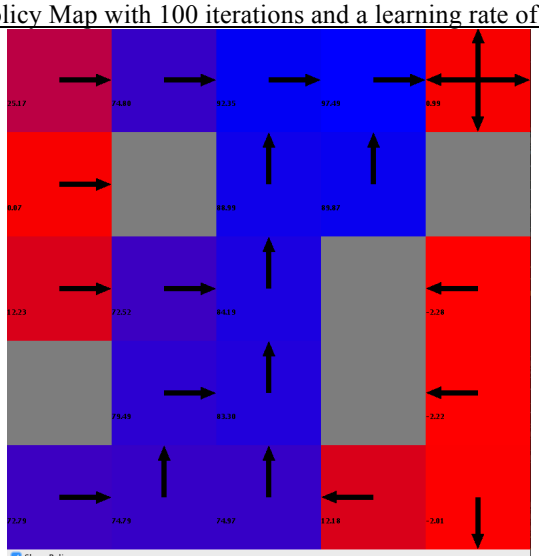


Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-300 Iterations)

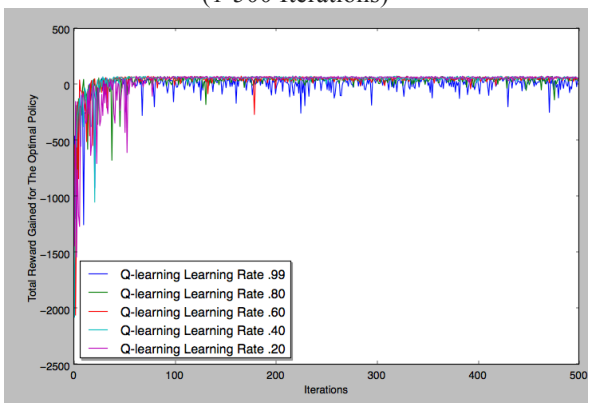


Inspecting the learning rates of .99 and .8, it is clear that they are not the best learning rates, I would say a learning rate of .2 works the best for this problem with a small number of states and seems to converge in around 100 iterations, but it still fluctuates a bit all the way up to 300 iterations, it was not as clear cut compared to when I ran policy and value iteration.

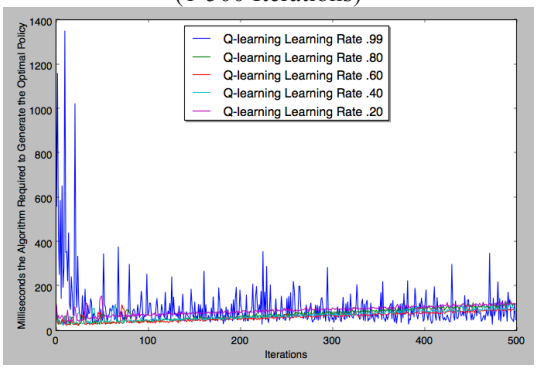
Policy Map with 100 iterations and a learning rate of .2



Total Reward Gained for The Optimal Policy
(1-500 Iterations)

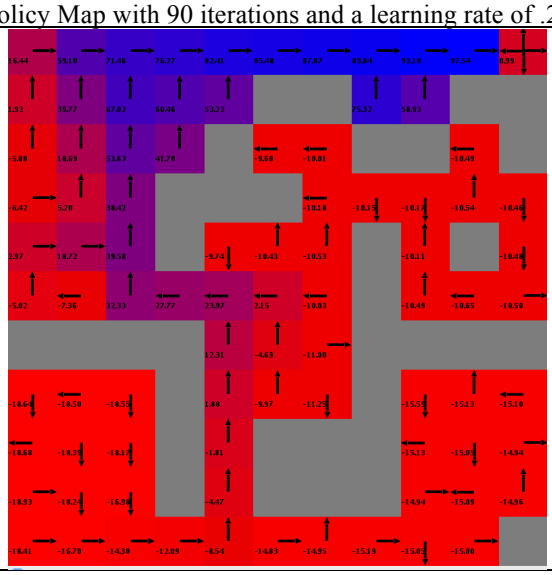


Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-500 Iterations)

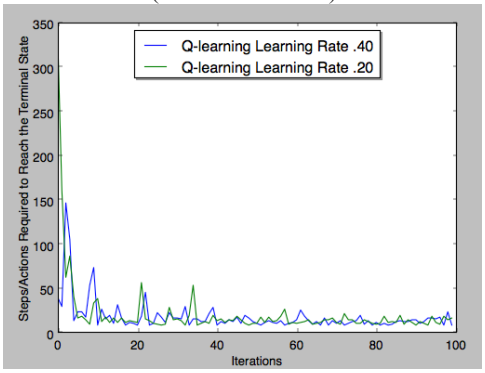


It is interesting that for the grid world with the larger number that the learning rate of .20 also preforms the best, it took more time than the other learning values except for a learning rate of .99, you can see that it got a lower number of steps to termination that was consistently lower than the other tested learning rates. Although none of them seem to have converged as well as policy or value iterations even with 500 iterations, they all still fluctuate pretty severely, but because the learning rate .20 seemed to fluctuate the least and seemed to sort of converge in 90 iterations.

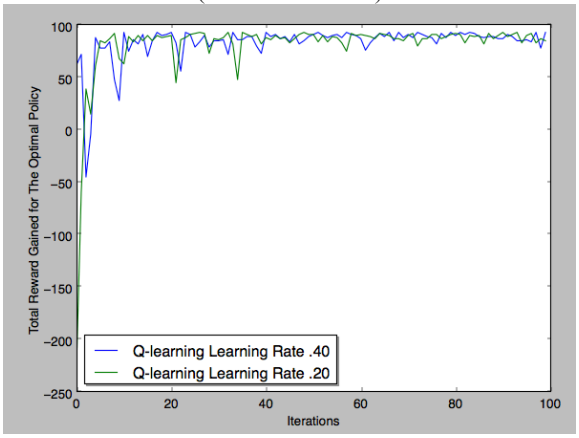
Policy Map with 90 iterations and a learning rate of .20



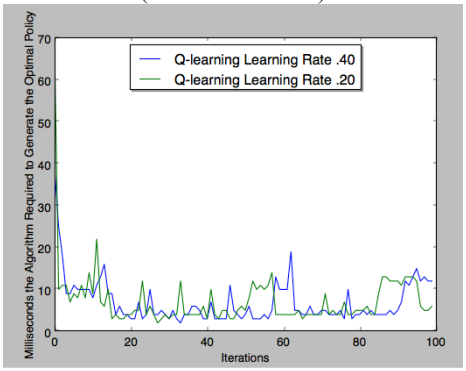
Steps/Actions Required to Reach the Terminal State
(1-100 Iterations)



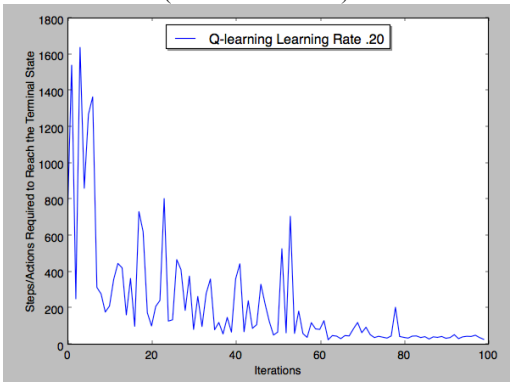
Total Reward Gained for The Optimal Policy
(1-100 Iterations)



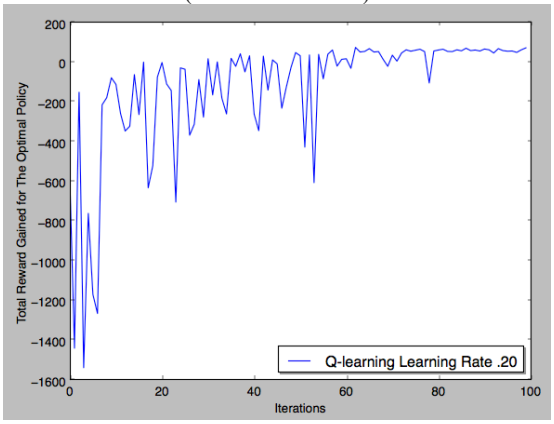
Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-100 Iterations)



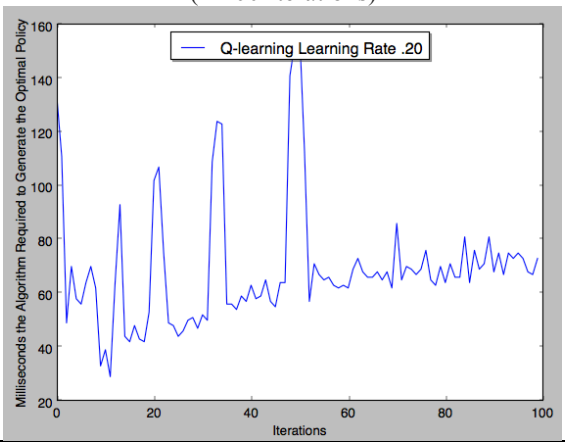
Steps/Actions Required to Reach the Terminal State
(1-100 Iterations)



Total Reward Gained for The Optimal Policy
(1-100 Iterations)



Milliseconds the Algorithm Required to Generate the Optimal Policy
(1-100 Iterations)



Conclusion

Grid World with Small # of States

Algorithm	Iterations Till Convergence	Steps/Actions	Time (Milliseconds)
Value Iteration	3	9	2
Policy Iteration	6	9	3
Q-Learning	100	9	12

Grid World with Large # of States

Algorithm	Iterations Till Convergence	Steps/Actions	Time (Milliseconds)
Value Iteration	18	29	22
Policy Iteration	16	27	30
Q-Learning	90	27	71

For both problems, value iteration and policy iteration work very well for finding an optimal policy, a draw back from using these they think the agent knows the reward and transitions for every step/action in the problem, also known as a form of domain knowledge. We can see that for the grid with a small number of states that value iteration converges to an optimal policy in only 2 milliseconds. The grid world with a large number of states, I thought that Q learning would get to a lower number of steps to find the best optimal policy (one with less steps/actions) although it is much slower than the rest of the algorithms. I think it was very interesting that for both grid world problems Q-learning preformed the worst in terms of time to convergence, and did not do as well as policy iteration in terms of time to convergence to optimal policy with the grid world with a large number of states. I believe this significant time difference using Q-learning in the small and large state problems was because Q-learning is a form of model-free learning where each iteration the agent goes through the environment and after each iteration the order of steps the agent took are stored for the next time the agent reruns the policy. This in turn takes much, much more computational time and iterations for convergence but in cases where the environment is bigger and more complex, although it can optimal policy than value or policy iterations, computational time is the tradeoff of this.