CS 7641 - Markov Decision Processes (Assignment 4)

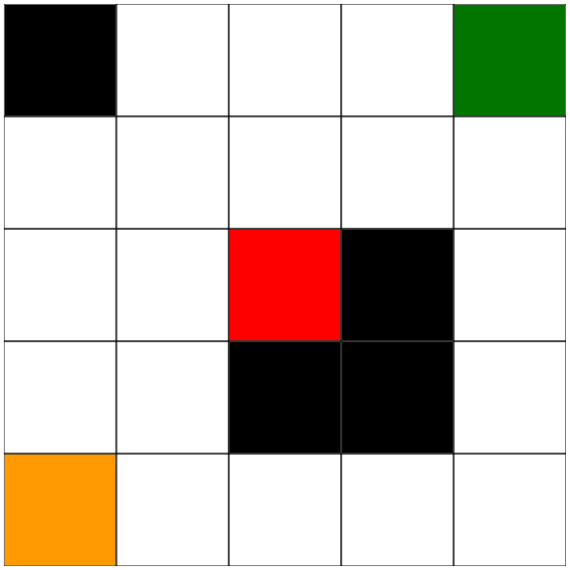
Introduction:

This paper explores three techniques in reinforcement learning that are used to make decisions. Two interesting Markov Decision Processes in Grid World are used to demonstrate the techniques. One Grid World has a small 5x5 grid of states, referred to as Easy Grid World, and the other Grid World has a large 15x15 grid of states, referred to as Hard Grid World. Each MDP is solved using value iteration, policy iteration, and Q-Learning. The results of the three techniques - iterations vs. time, convergence, steps, and reward, will be compared and contrasted with each other. The algorithms are all implemented using the BURLAP java code library in Jython.

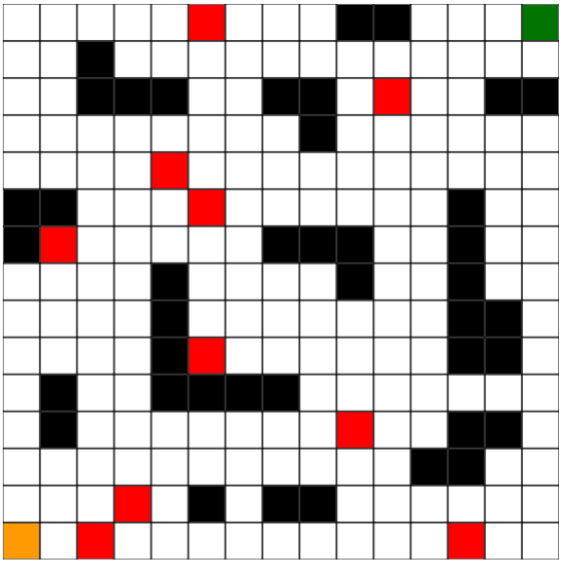
Description of Grid World:

Grid World is a dynamic environment that is popular in Reinforcement Learning because it's easy to visualize whether the solution is optimal or not, visualize walls or negative rewards, and compare policies. Easy Grid World has 5x5 = 25 distinct states and Hard Grid World has 15x15 = 225 distinct states. The black squares indicate a wall that cannot be moved into. The orange square in the bottom left cell is the start state at (0,0). The green square is the terminal state at (n, n) that has a reward of 100. The white cells have a reward of -1 and the red squares have a reward of -10. The agent can take 4 different actions (up, down, left, right). It moves in the correct direction 80% of the time and in each of the incorrect directions 6.67% of the time. The goal is to find the optimal policy that maximizes the future discounted reward or arrive at the terminal state optimally.

5x5 Grid World



15x15 Grid World



Why is Grid World interesting?

The Grid World problem is interesting because there are many real life applications for it. Many things in our society can be represented as a grid. Traffic is referred to as grid locked sometimes and city planning requires grids, such as power line grids. Grids are also useful for inventory or logistics planning. Even board games like Chess and Checkers utilize a grid. Solving these problems can help with practical problems that people face in the real world by solving them optimally in less iterations, steps, time, and with more reward.

Description of the techniques used:

Policy Iteration:

Policy iteration starts with some initial policy by taking an action to the neighboring state that maximizes the long-term expected reward. After each iteration, policy is improved by taking the best action given the updated expected

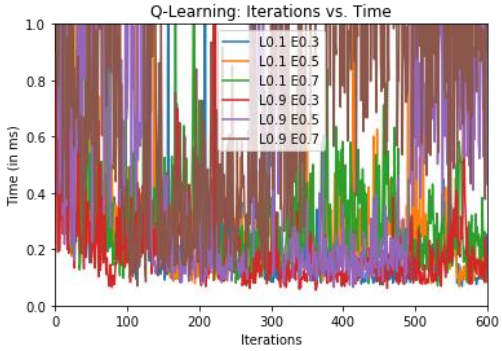
utility at each state. Policy iteration always converges. Policy iteration sometimes takes less iterations to converge than value iteration because once the order of the utilities are achieved, the optimal policies can be found.

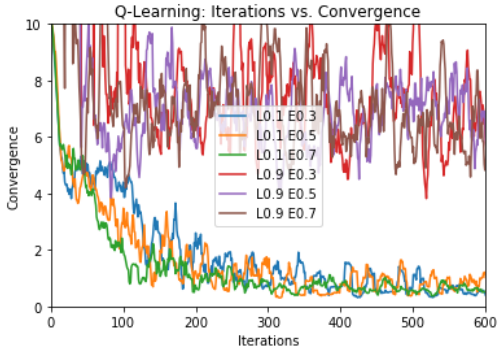
Value Iteration:
Value iteration uses Bellman's equation to update the utilities of states until there is convergence. Starting with arbitrary utilities, after each iteration it updates utilities based on all neighboring state rewards it can reach and possibly itself if the action runs into a wall. The utility of a state is the immediate reward plus discounted future rewards. It repeats until convergence and is guaranteed to converge.

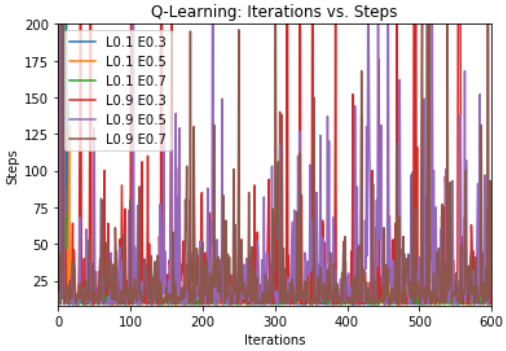
Q-Learning:
Q-learning uses transitions (data) to directly produce the solution to the Q equations. This is a model-free learner that derives optimal policy through iterations and interactions with the environment because it doesn't know the state transitions or the rewards. The agent only knows the possible states and actions and can observe the current state as it interacts with it. Through incremental learning of immediate and delayed rewards, Q-learning converges on the expected average and optimal solution. Also, like simulated annealing, Q-learning trades off between exploration and exploitation, sometimes transitioning to a random Q hat with probability epsilon.

Easy 5 x 5 Grid World:

Q-Learning Hyperparameter Selection:
To compare the three algorithms, I will first perform hyperparameter selection to find the best Learning Rate and Epsilon for Q-Learning. The Learning Rate trades off between learning nothing, alpha = 0, and full learning, alpha = 1 . The Epsilon is a greedy approach that takes a random action, Q hat, with probability Epsilon. A learning rate of 0.1 and 0.9 and epsilon values of 0.3, 0.5, and 0.7 are evaluated.

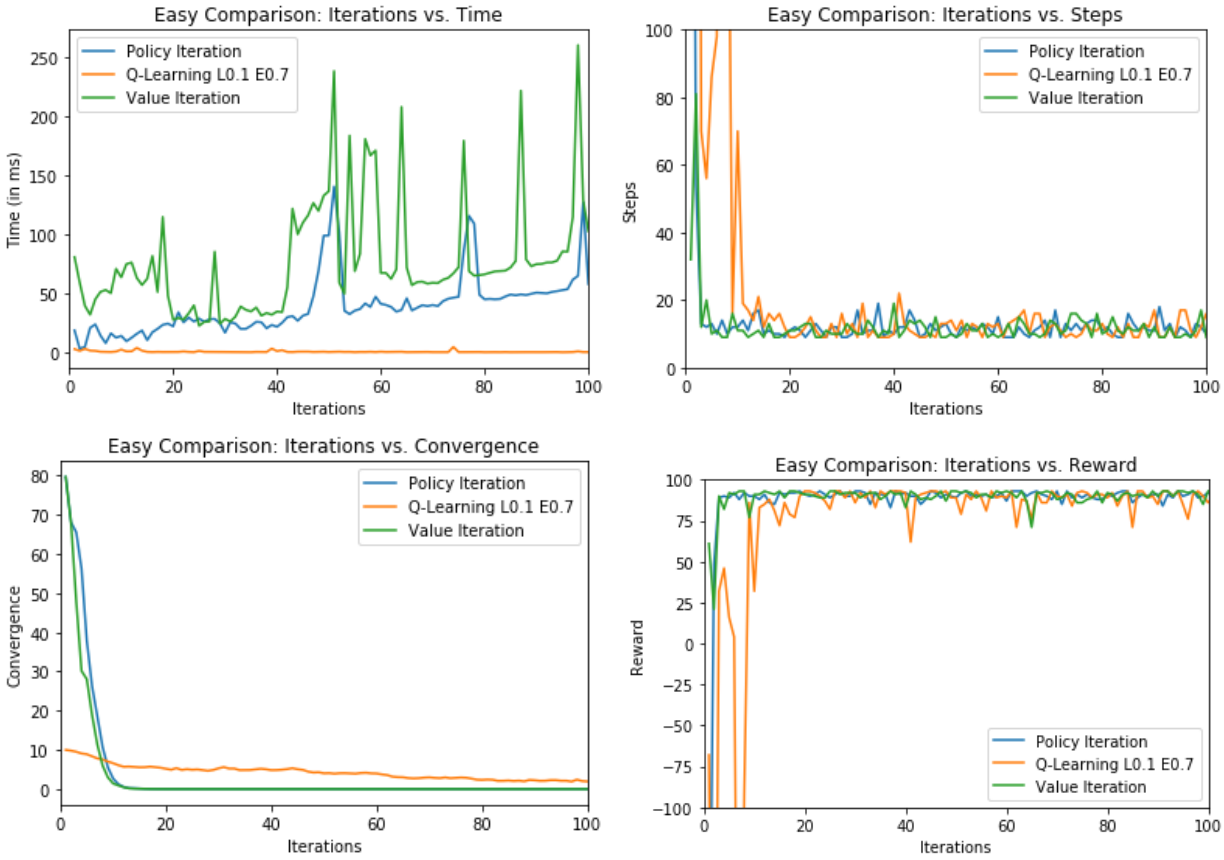






The best parameters chosen for Q-learning is: learning rate = 0.1 and an epsilon value = 0.7. This means that the agent will pick a random action or explore with probability 0.7 (Exploration helps find better policies) and does not trust learning as much. This value is chosen because L0.1 E0.7 has one of the lower wall clock times listed in green and the lowest convergence value that has an elbow point around 100 iterations. The step time also doesn't fluctuate as much as the L 0.9 values, which is good. The convergence value for Q-learning measures the average of the last 10 Q-value changes that occurred in the last learning episode. Q-learning with parameters learning rate = 0.1 and epsilon value = 0.7 will be used to compare with Value Iteration and Policy Iteration.

Value Iteration, Policy Iteration, and Q-Learning Comparison:
A discount rate of 0.99 is used for all three algorithms to discount future rewards and estimates.



Policy Iteration and Value Iteration converges around 15 iterations while Q-Learning starts to converge around 500 iterations (not shown here).

Policy Iteration and Value Iteration converge around the same number of iterations, but the clock time (in milliseconds) differs in significantly. At almost every iteration, based on the time chart, Policy Iteration is twice as fast as Value Iteration. This makes sense because in policy iteration, once the order of the utilities are achieved, optimal policies can be found. The time for Q-learning is constant and very low compared to the other two, due to its exploring nature with epsilon = 0.7.

Q-Learning does not converge to the same answer in as many iterations. It converges to a low value around 500 iterations compared to what Policy Iteration and Value iteration can achieve in about 12 iterations. However, the tradeoff and benefit of Q-learning is the constant low clock time (~1 ms).

Compared to Value Iteration and Policy Iteration, Q-learning converges at much later iterations, but can compete in terms of Reward and Steps vs. Iterations with similar values above 20 iterations. Q-learning performs much better in computational time, which makes it worth the time vs iterations tradeoff.

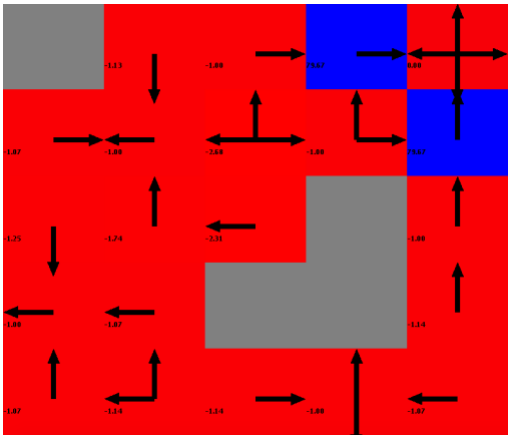
Hyperparameter tuning of exploration strategies in Q-learning found that a high epsilon = 0.7 benefited the algorithm more and finds a better policy compared to more exploitation. Also, a low learning rate = 0.1 achieves much better convergence than a learning rate = 0.9, they produce convergence values of ~1 and ~8, respectively.

CS 7641 - Markov Decision Processes (Assignment 4)

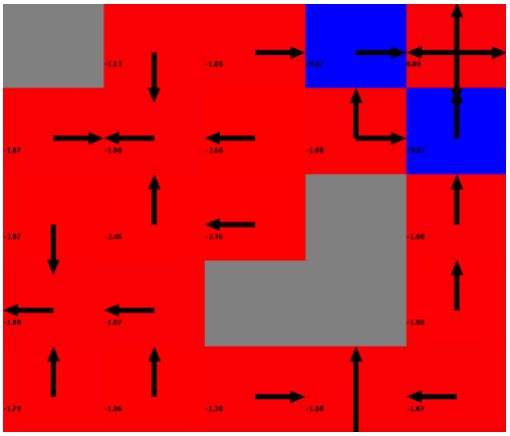
Looking at the Utility and Policy Map:

1 iteration:

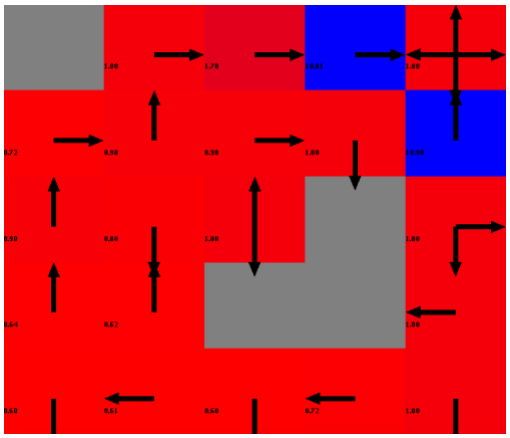
Value = 1



Policy = 1



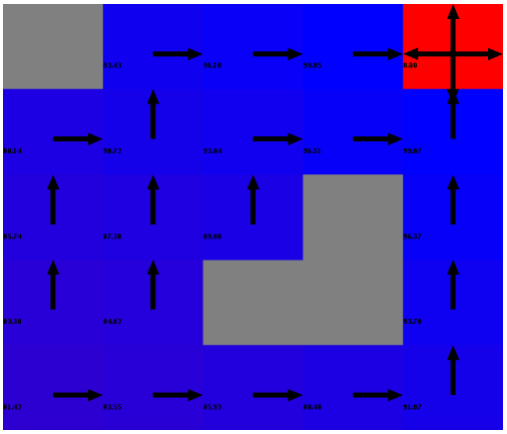
Q-Learning = 1:



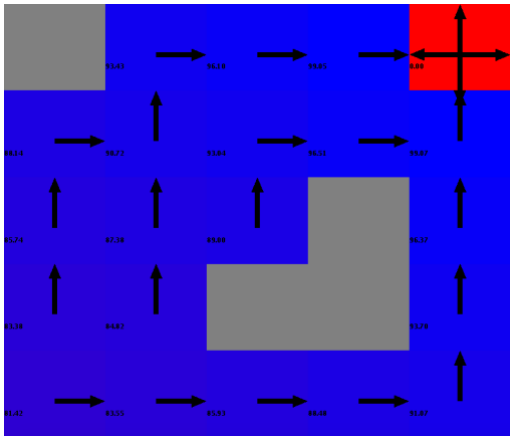
At the first iteration, all of the utilities and policies differ between the three algorithms.

100 Iterations:

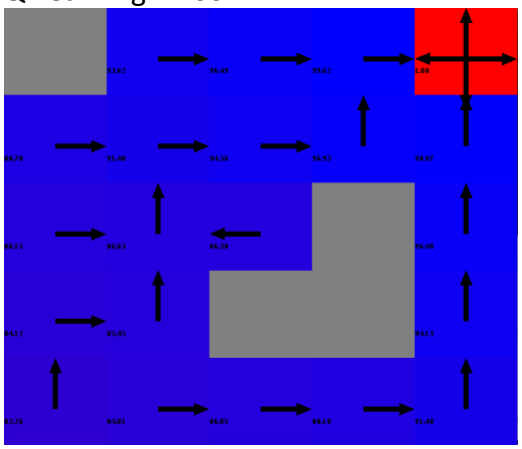
Value = 100



Policy = 100:



Q-Learning = 100:



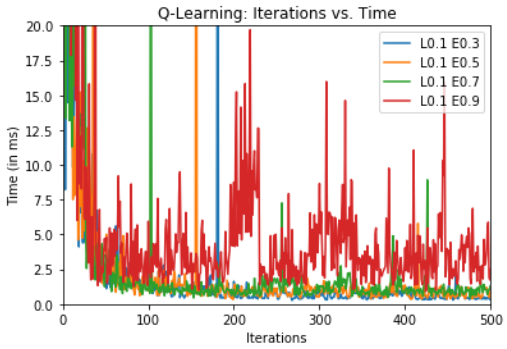
After 100 iterations, Value Iteration and Policy Iteration produce the same exact utility and policy values. This is because they already converged at a much lower iteration, around 12. Q-Learning doesn't really converge at 100 iterations and the policy map has several differences with Policy and Value Iteration. The policy arrows in Q-Learning change directions at a higher iteration amounts, not shown here. However, the utilities are somewhat close due to the grid being small.

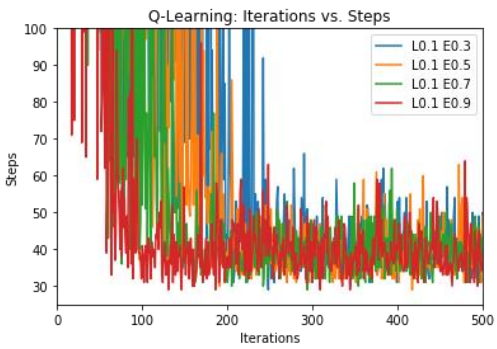
Hard 15 x 15 Grid World:

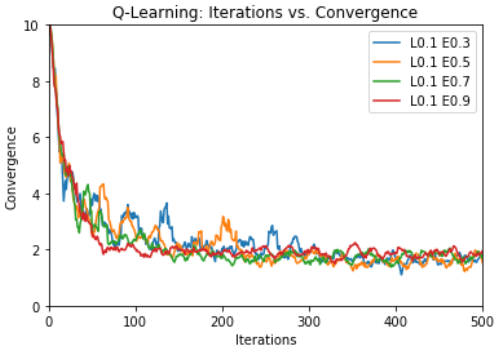
Q-Learning Hyperparameter Selection:

Again, I will first perform hyperparameter selection to find the best Learning Rate and Epsilon for Q-Learning. The Learning Rate trades off between learning nothing, alpha = 0, and full learning, alpha = 1 . The Epsilon is a greedy approach that takes a random action, Q hat, with probability Epsilon. A learning rate of 0.1 and epsilon values of 0.3, 0.5, 0.7, and 0.9 are evaluated.

Q-Learning Hyperparameter selection for Learning Rate and Epsilon.



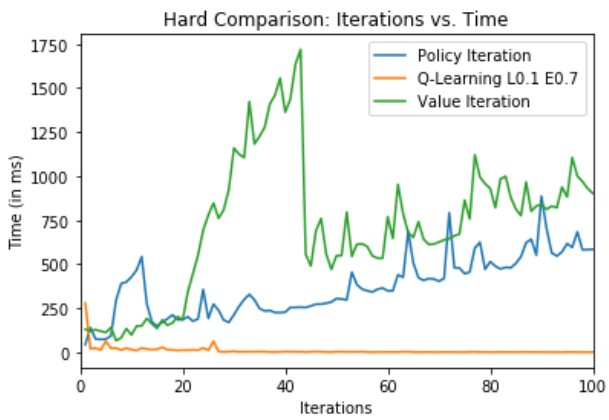


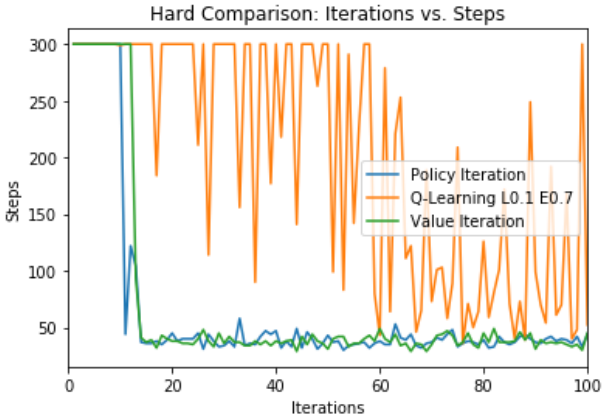


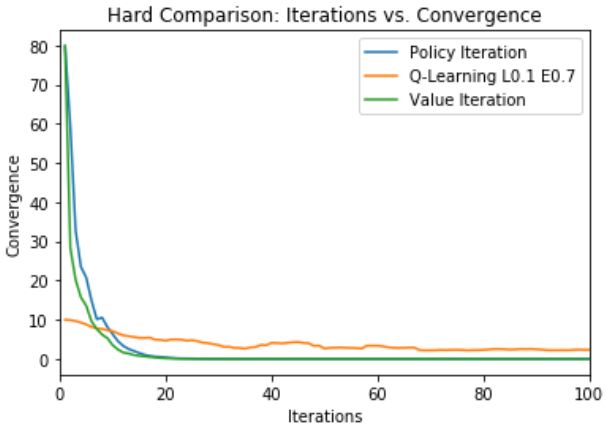
The best parameter for learning rate = 0.1 is an epsilon value = 0.7. This means that the agent will pick a random action or explore with probability 0.7. This value is chosen because Epsilon = 0.7 has one of the lowest wall clock times along with 0.5. It's also one of first values to settle around 30-50 steps. Epsilon = 0.7 also has an elbow point in the convergence chart around 200 iterations. The convergence for Q-learning measures the average of the last 10 Q-value changes that occurred in the last learning episode. Learning rate = 0.1 and epsilon value = 0.7 will be used to compare with Value Iteration and Policy Iteration.

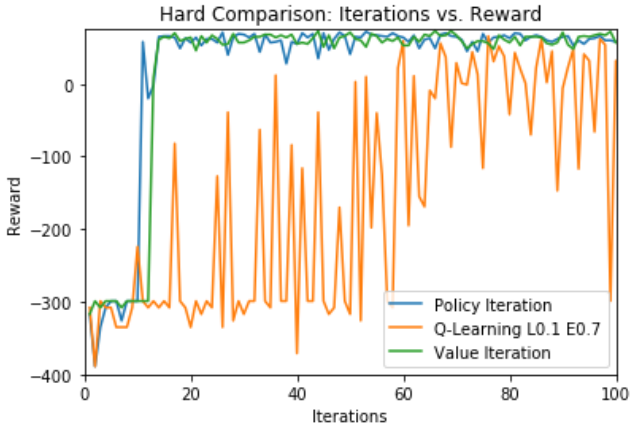
Value Iteration, Policy Iteration, and Q-Learning Comparison:

A discount rate of 0.90 is used for all three algorithms to discount future rewards and estimates.









Policy Iteration and Value Iteration converges around 18 iterations while Q-Learning starts to converge around 600 iterations (not shown here) with a value of ~1.72.

Again, Policy Iteration and Value Iteration converge at much lower iterations, around 18, but Q-learning is much faster. Policy Iteration is more computationally time efficient due to the algorithm's linear nature compared to the non-linear utility in Value Iteration. The tradeoff in time of 600 iterations of Q-learning, Q-learning 600 x 1ms = 600 ms, is almost faster than just a few iterations of Policy and Value Iteration. However, the convergence for Policy and Value iteration is much better.

The number of states affects the step size for all of the techniques significantly. The clock time for Value Iteration and Policy Iteration increases exponentially, while Q-learning still remains constant. Also, the Reward vs. Iteration is poor at earlier iterations for Q-Learning vs. Policy and Value Iteration.

Compared to Value Iteration and Policy Iteration, Q-learning converges at much later iterations, but has similar values for Reward and Steps vs. Iteration for values above 80 iterations. Q-learning performs much better in computational time, which makes it worth the tradeoff.

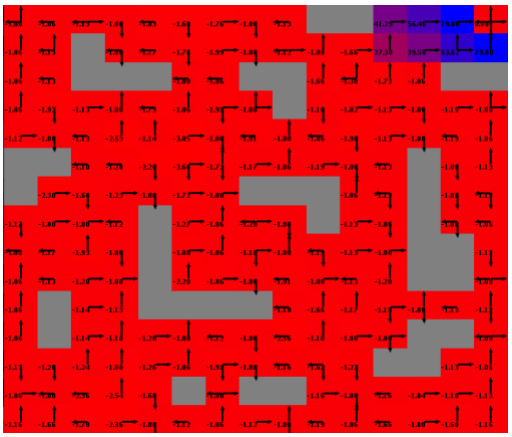
Hyperparameter tuning of exploration strategies in Q-learning found that a high epsilon = 0.7 benefited the algorithm more in finding a better policy compared to more exploitation. Also, a low learning rate = 0.1, achieves good convergence, which is the same as in Easy Grid World.

CS 7641 - Markov Decision Processes (Assignment 4)

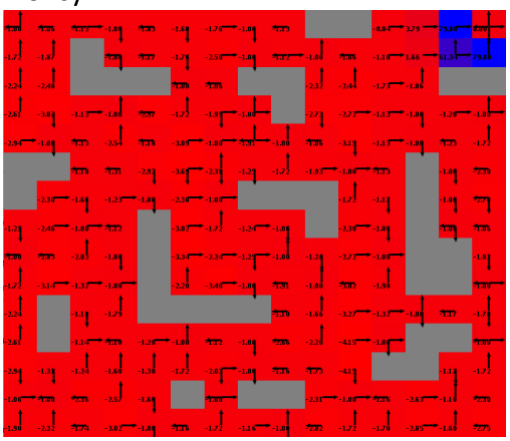
Looking at the Utility and Policy Map:

1 iteration:

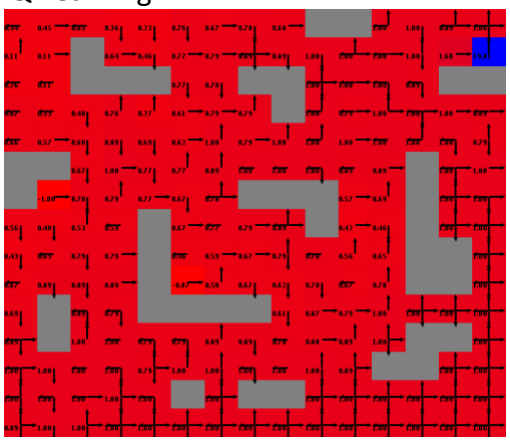
Value = 1



Policy = 1



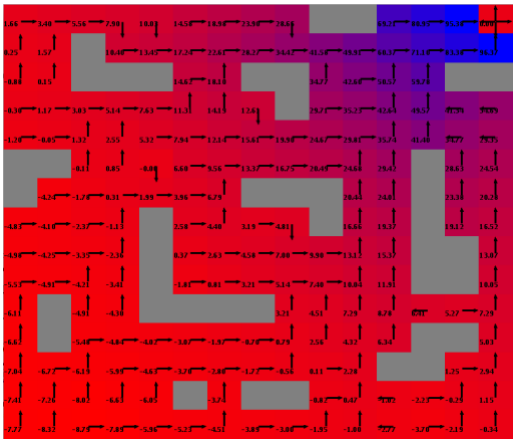
Q-Learning = 1



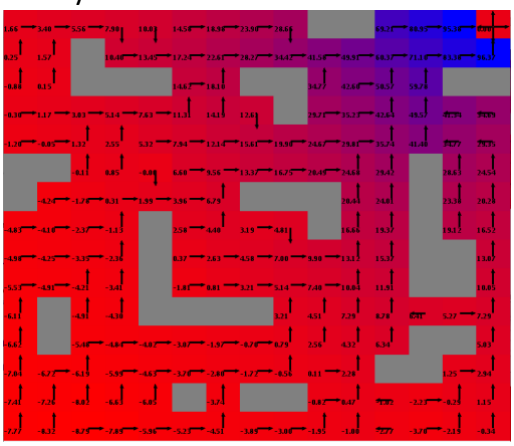
At the first iteration, all of the utilities and policies differ between the three algorithms.

100 iterations:

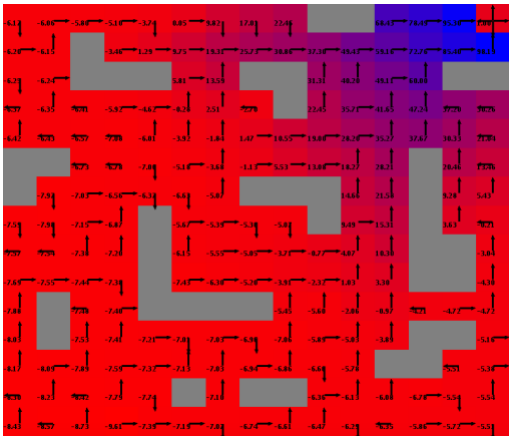
Value = 100



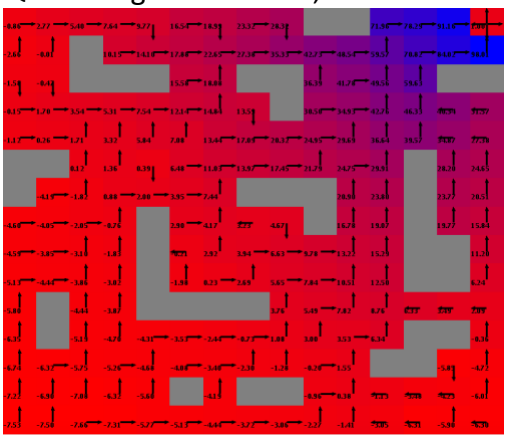
Policy = 100



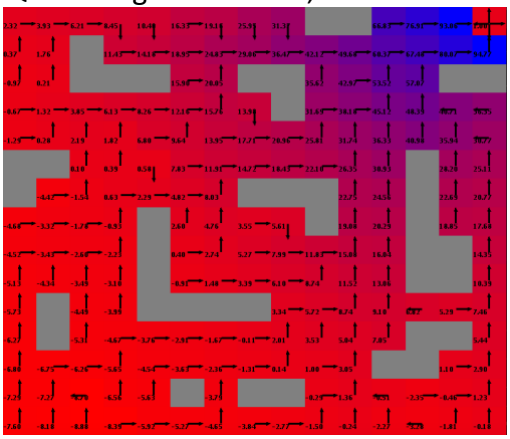
Q-Learning = 100



Q-Learning Iteration = 1,000



Q-Learning Iteration 10,000:



After 100 iterations, Value Iteration and Policy Iteration produce the same exact utility and policy values. This is because they already converged at a much lower iteration, around 18. Q-Learning doesn't really converge at 100 iterations and the policy map has several differences with Policy and Value Iteration. The policy arrows in Q-Learning change directions at a higher iteration amounts of 1,000 and 10,000.

Conclusion:

With a higher number of states, the time that it takes Q-Learning to learn is constant, while it increases exponentially for Value Iteration and Policy Iteration. The step size for all of the techniques increases as the number of states increases. The clock time for Value Iteration and Policy Iteration increases exponentially with increased states or harder grid world problems, while Q-learning time still remains constant. Policy and Value Iteration guarantees convergence when Q-learning sometimes doesn't. Also, the Reward vs. Iteration is poor for Q-Learning compared to Policy and Value Iteration. Q-learning sometimes becomes much harder in large problems because exploration is much more difficult with more states to explore. In harder problems with more states, Q-learning performs poorly, while Value Iteration and Policy Iteration still perform well. The trade-off between good clock time with Q-learning and convergence with Policy and Value Iteration is something to take into consideration when using these techniques.