Jeremy Kang
Jeremy.kang@gatech.edu
CS 7641: Machine Learning – Assignment 4

Introduction

Two Markov Decision Processes (MDP) problems – one with a 5x5 state space and another with an 11x11 state space – were solved using policy iteration, value iteration, and Q-Learning. The results are examined and compared below.

Software and Approach

Brown-UMBC Reinforcement Learning and Planning (BURLAP) Java code library was used to solve the MDP problems and Microsoft Excel was used to analyze and graph the results.

The actions were north, east, south, or west with a probability of 80% of succeeding, 10% of using the epsilon greedy policy, and 10% of using a random action. Rewards were set to -1 and the goal state was set to 100. This approach makes it so that the agent does not collect positive rewards on its path until it reaches the goal, encouraging the agent to reach the terminal state as quickly as possible. The number of iterations was also capped at 100 for simplicity of analyzing the two problems. For Q-Learning, the epsilon greedy policy was set to 0.1.

Problems

5x5 Grid World Problem

A 5x5 Grid World problem is a relatively easy MDP problem with, at most, 25 states. There are four states that represent obstacles/invalid states that the agent cannot be in which totals to 21 possible states. The main objective for the agent is to get from the starting state (1, 1) to the goal state, (5, 5). There are two main routes (or utilities) for the agent to take: going north and then east, or going east and then north. This particular problem is interesting because of the fact that there exist only two optimal paths. Solving the problem with policy iteration, value iteration, and Q-Learning may reveal different choices that these methods make.

11x11 Grid World Problem

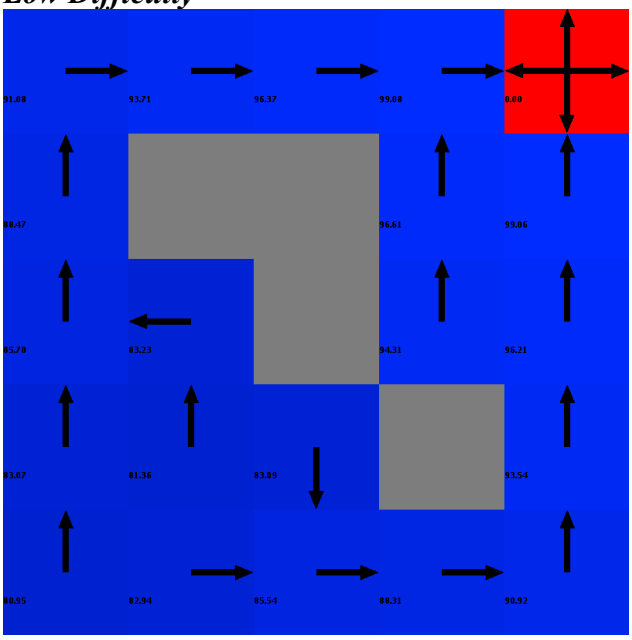
Compared to the 5x5 Grid World problem, the 11x11 state space has more than 4 times the amount of states at 93 (including 28 invalid states). Although the objective is the proportionally the same where the agent starts at (1,1) and the goal state is at (11, 11), there are significantly more routes for the agent to take which makes for a far more interesting problem.

Both of these problems are akin to the problem of self-driving cars where they must avoid obstacles, but still try to reach their destination with an optimal policy. Of course, an even more realistic problem would be to have multiple agents that also navigate throughout the given world. However, not only does that raise additional problems of navigating taking into account other agents, but also the number of states would increase by n^2 where n = number of agents. For the purposes of this assignment,

these were not included, but should be further explored to see how it affects the sequences of the policy iteration, value iteration, Q-Learning utility.

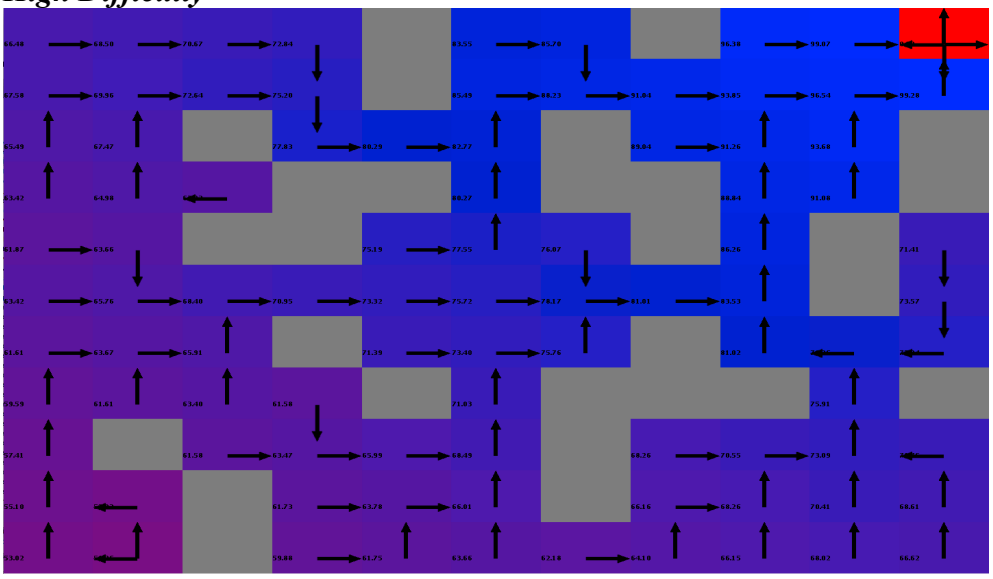
Policy Iteration

Low Difficulty



As shown above, Policy Iteration suggests an optimal policy of either going north and then east, as well as going east and north. Both of these utilities are intuitively correct, as the obstacles/invalid states do not have any bias towards one policy over the other. The most interesting state is at (2, 2) in that any direction would be valid in an optimal policy chosen by any of the methods. However, this was not indicated by the Policy Iteration so differences may be seen with either Value Iteration or Q-Learning.

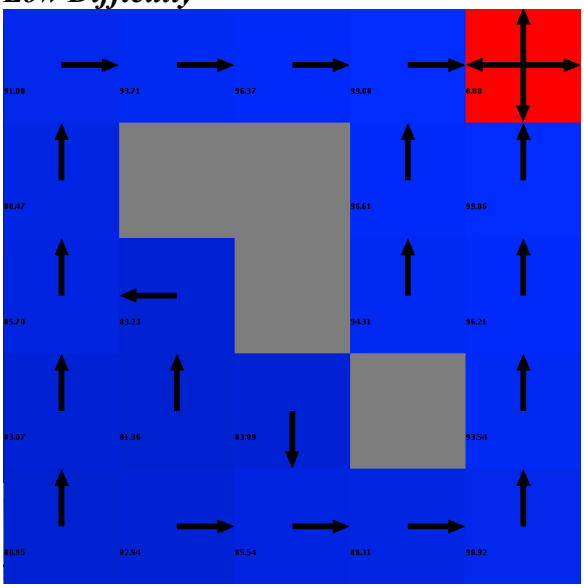
High Difficulty



The high difficulty shows a far more interesting policy map. There are multiple routes the agent may take and is also shown by the graphs below. The majority of the states-actions are expected. However, one of the interesting states is (2, 1) where either west or north is the suggested action. Again, this makes sense because these states are available to take an action in, but the most interesting aspect of this point is that two actions were not suggested in the low difficulty problem for the (2, 2) state.

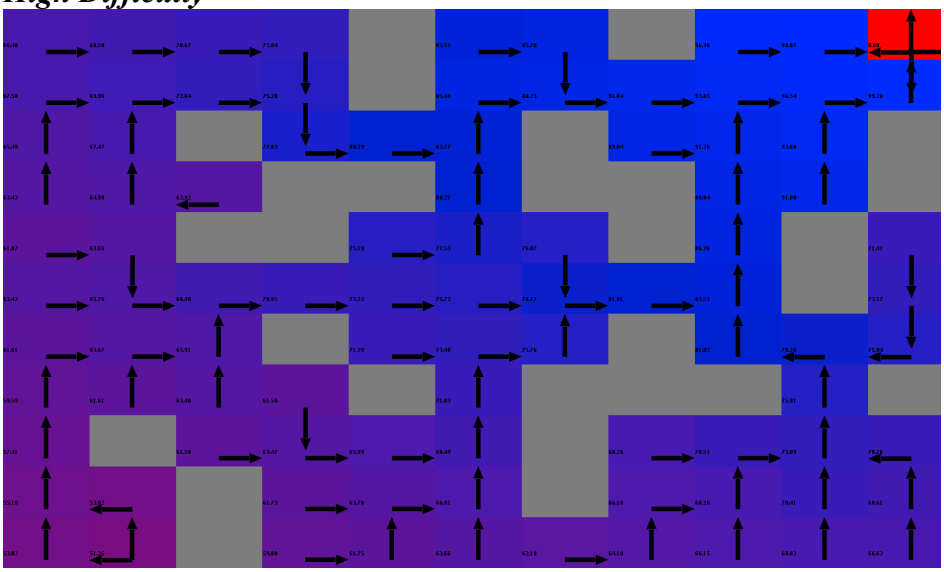
Value Iteration

Low Difficulty



Value Iteration's policy was the same as the one found for Policy Iteration. This isn't surprising as the problem itself is relatively easy. Two routes, both with valid and optimal utilities are available for the agent to choose from.

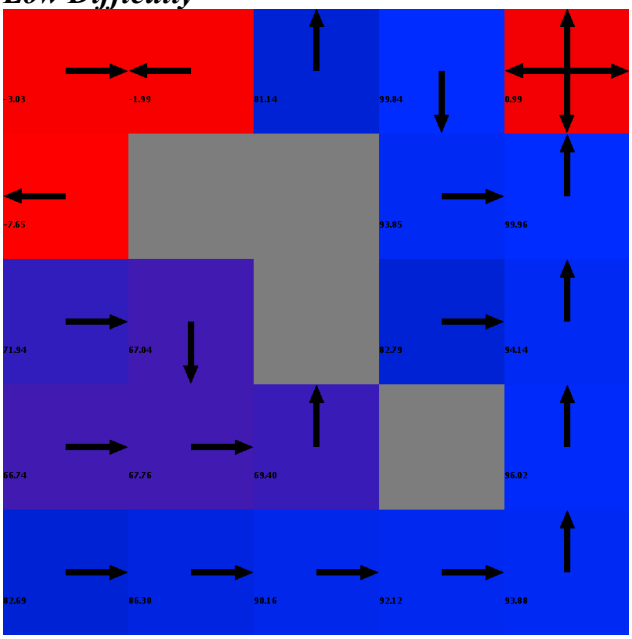
High Difficulty



Again, Value Iteration found the exact same policy as Policy Iteration.

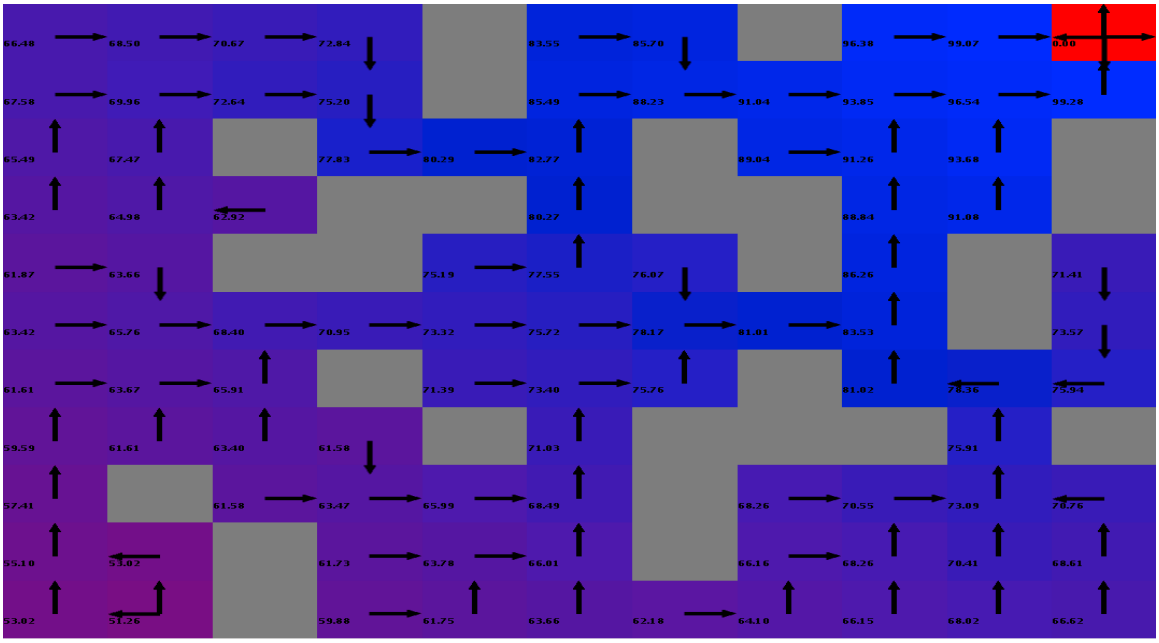
Q-Learning

Low Difficulty

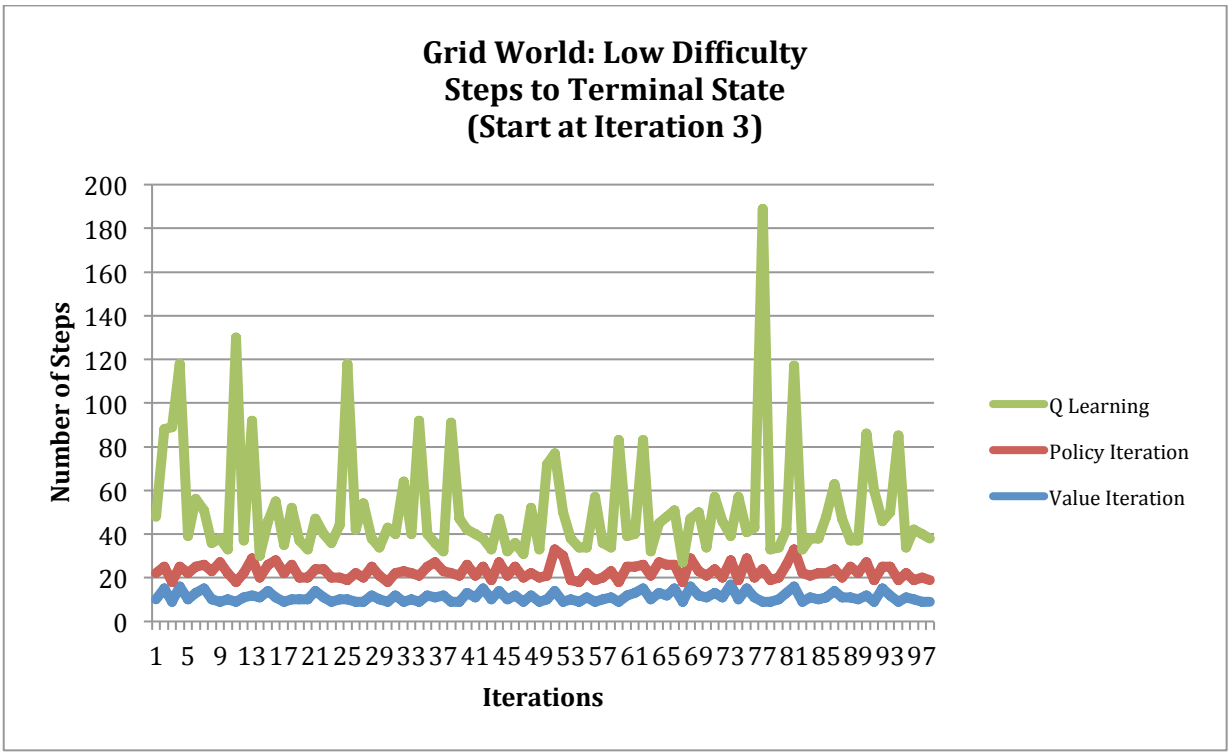


Q-Learning showed an interesting behavior for the top left quartile of the grid. The action chosen if the agent was in the (1, 4), (1, 5), (2, 5) spots appeared counter-intuitive. Re-running the algorithm showed that in some cases, the other utility (path going east and then north) had the same problem.

High Difficulty

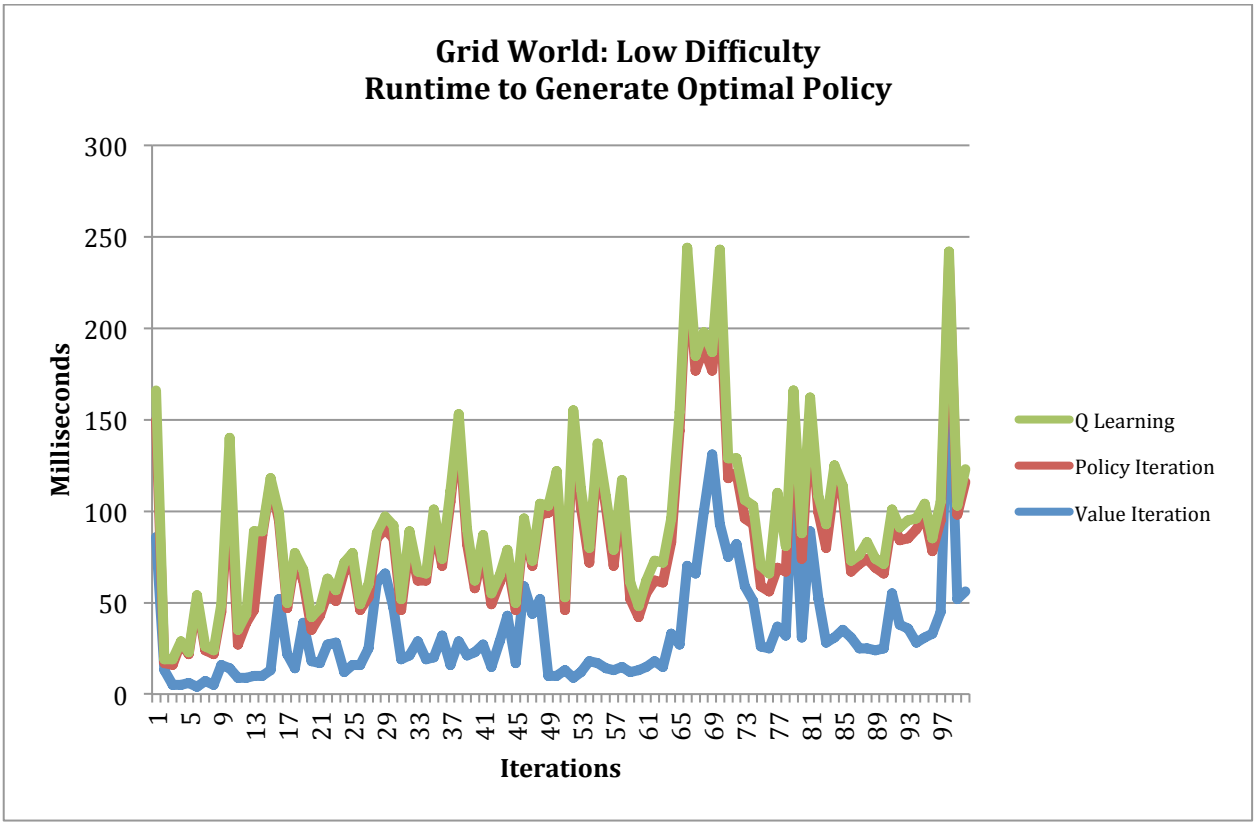


Graphs: Low Difficulty



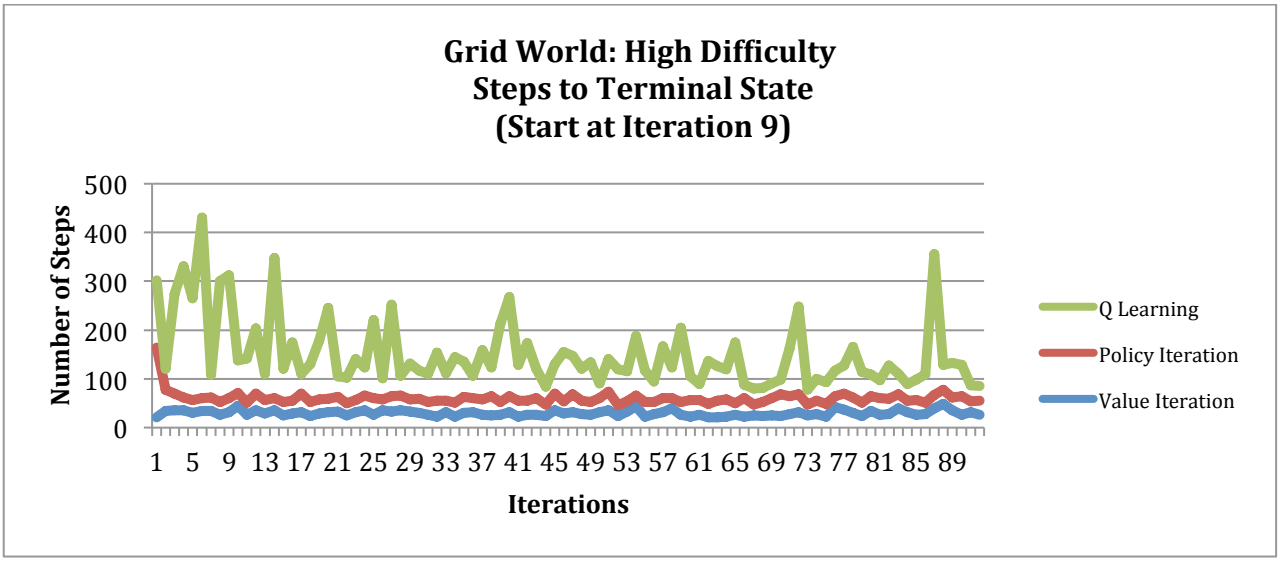
Policy and Value Iteration appeared to converge within a relatively low number of experiments. However, Q-Learning shows irregular number of steps across the 100 iterations, suggesting that it may be stuck in local optima and may benefit with further iterations.

Note that the graph shows the methods starting at iteration 3 so that the results were better viewed. This is because for Value Iteration, iteration 1 took 496 steps and iteration 2 took 768 steps. Policy Iteration and Q-Learning also had similar erratic behavior for the first two steps.



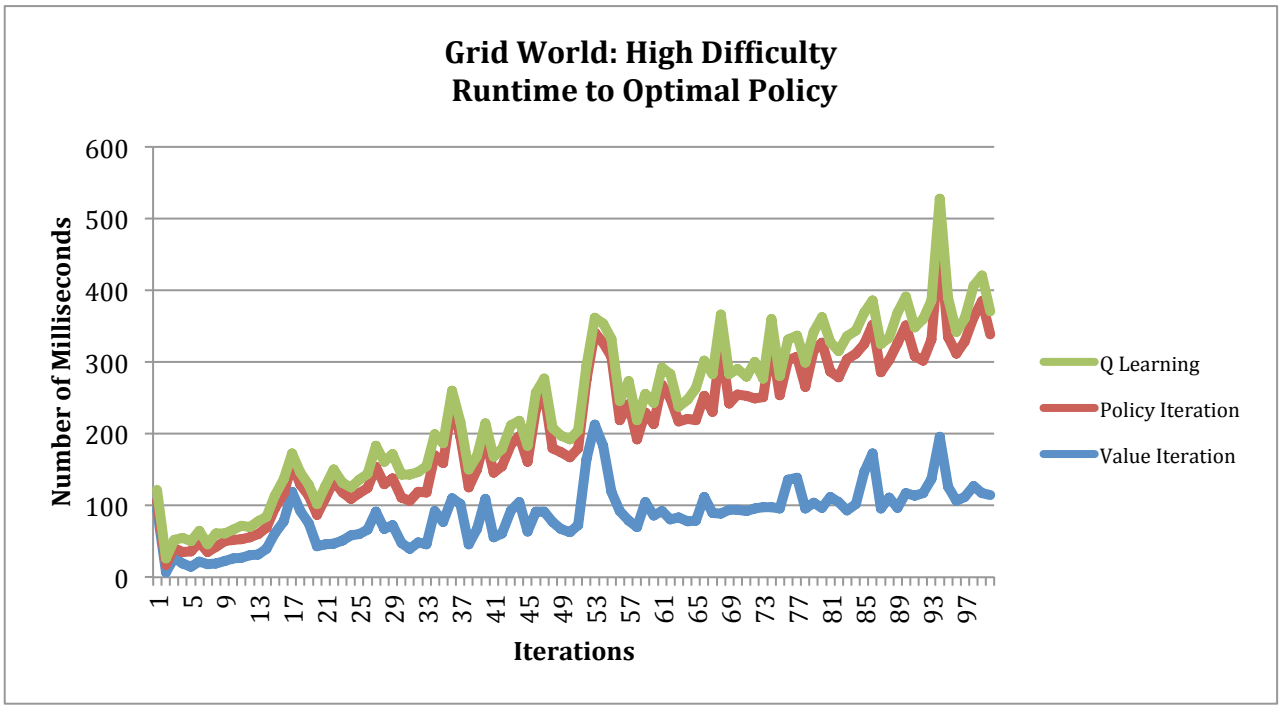
Value Iteration required the least amount of time to get the optimal policy. However, one interesting behavior was that the Policy Iteration and Q-Learning methods appear very similar in runtime.

Graphs: High Difficulty



Although the erratic behavior persists for Q-Learning, the number of steps required to reach the terminal state appears to be slowly going down. This suggests that Q-Learning may have converged within the 100 iterations for the High Difficulty problem.

Similar to the graph for the low difficulty grid world, the graph begins at iteration 9 because of a higher step count for the first 8 iterations (Policy Iteration took up to 12,458 steps before converging to an average of 633.33 steps across 100 iterations).



Value Iteration performed the best again and appears to remain relatively stable across iterations. However, Q-Learning and Policy Iteration, again, performed very similar. However, the runtime appears to perform in a linear fashion in which when the number of iterations increases, the longer it takes to converge. This may indicate that Policy Iteration may be stuck in local optima as well.

Conclusion

As depicted in the policy maps and the graphs, Value Iteration converged in less iteration and in less time for both the high and low difficulty problems. Policy Iteration also appears to converge for both problems. However, the runtime is very similar to Q-Learning for both problems. It is important to note that after running the algorithms multiple times, there were times that Policy Iteration performed the best. However, the majority of the time, Value Iteration was the best performing method, on average.

On the other hand, Q-Learning takes the most time and steps in order to reach the terminal state. This may be explained in Q-Learning's erratic behavior across 100 iterations, suggesting that it may be stuck in local optima. Unfortunately, more iterations

were not tested. Changing some exploratory or exploitatory factors may lead to better performance for Q-Learning. Specifically, if Q-Learning was allowed to estimate the state space and then ‘exploit’ or use that knowledge to navigate through the space, it could have performed better or, at the very least, converged in 100 iterations.

There are many parameters that can be configured to see what changes happen to the state space. One major change could be the addition of the ability to move diagonally across the world. This would decrease both runtime and the number of steps taken to reach the terminal state. Adversarial agents or obstacles were excluded from both MDP problems and would also significantly change the policies chosen by the policy iteration, value iteration, and Q-Learning methods.