James Corwell

Homework 4

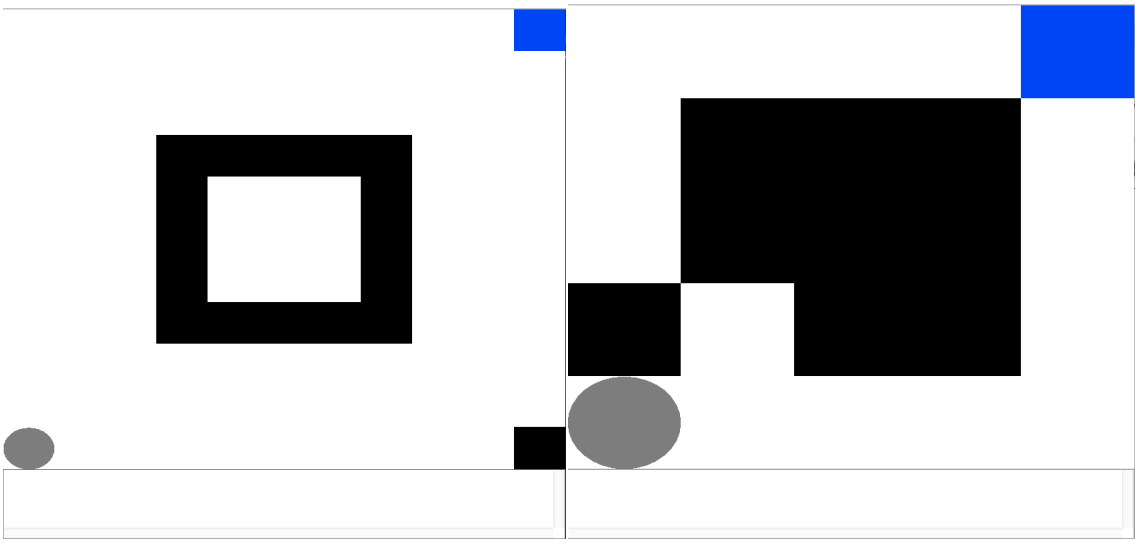
Markov Decision Processes

OMSCS CS 7641

I will implement a grid world path planning example for this paper using an example with a high number of states, as well as an example with a small number of states. These are interesting examples because path planning is a quintessential problem in AI, and is taught in many different ways in a graduate CS program. Thus far in OMSCS I've encountered binary search, A-star, Dynamic Programming, and now MDPs. They're also interesting because it gives practical experience working with the problems presented to us in the lectures.

It is possible to visualize the following two grid worlds thanks to Burlap; they are as follows:

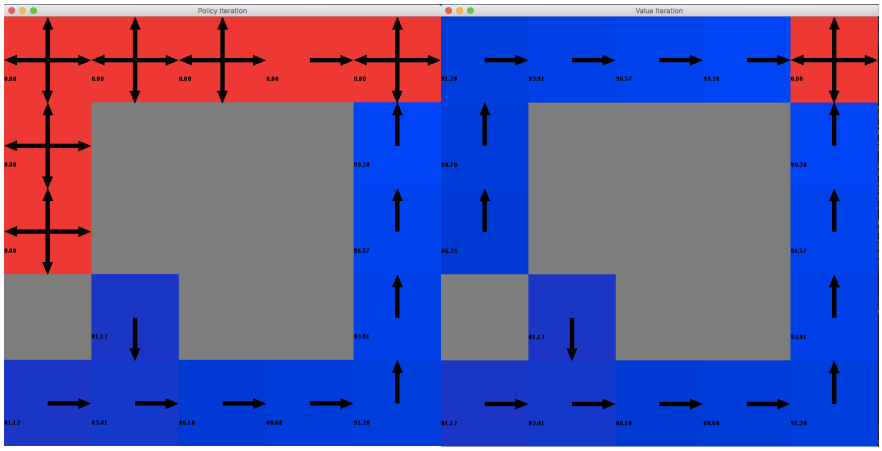
(Left:Hard Grid World (HGW), Right Easy Grid World(EGW))



The circle represents the moving object and the blue square is the goal.

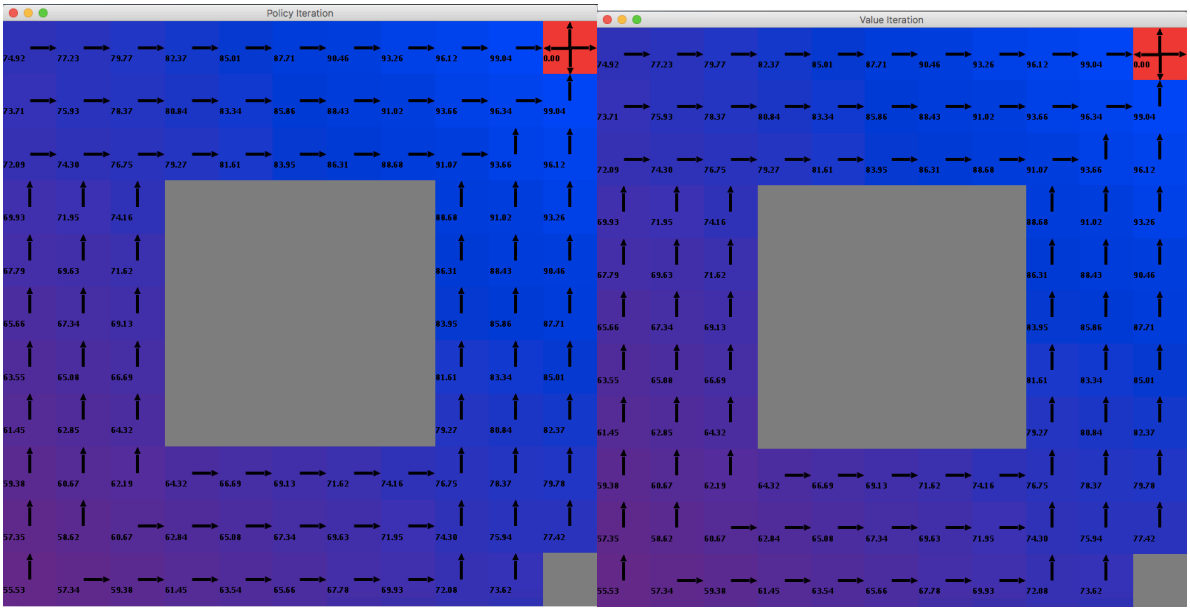
I then implement value and policy iteration on the grid world which generates the following policies:

Easy Grid World Policies



(left: policy iteration, right: value iteration)

Hard Grid World Policies

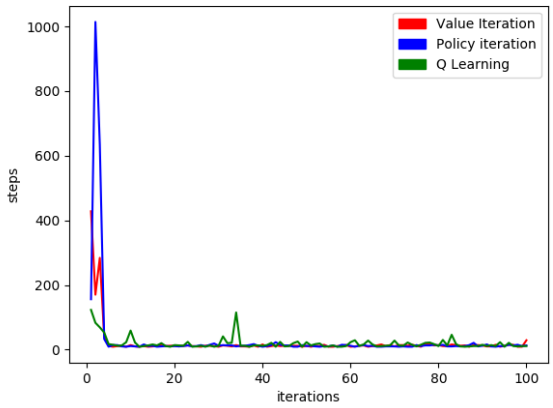


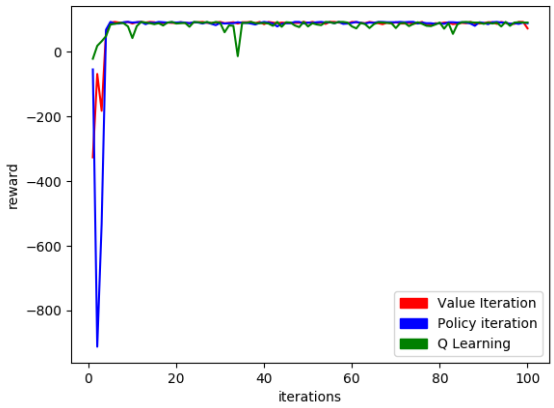
(left: Policy iteration, right: Value iteration)

It is notable that the EGW produced different plots for the Value iteration and Policy iteration algorithm, while the HGW produced the same policies.

The following plots are described as such for the Easy and Hard grid worlds; the first graph shows the number of steps (or actions) necessary to reach the terminal state versus the iterations. The second chart maps the reward for the optimal policy versus the number of iterations.

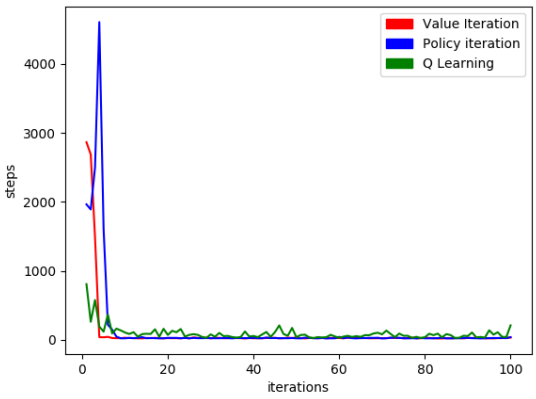
Easy Grid World Reward/Steps

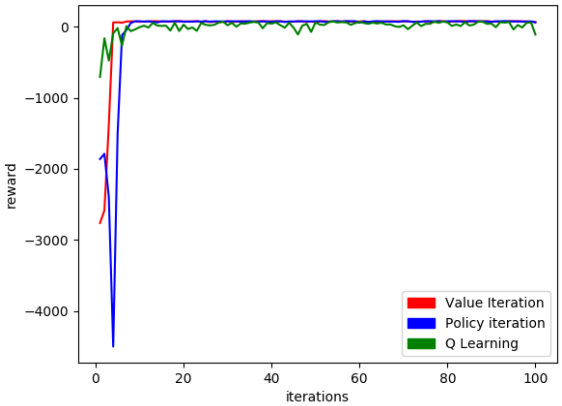




(Right: Reward/Iteration) (Left:Steps/Iteration)

Hard Grid World Reward/Steps



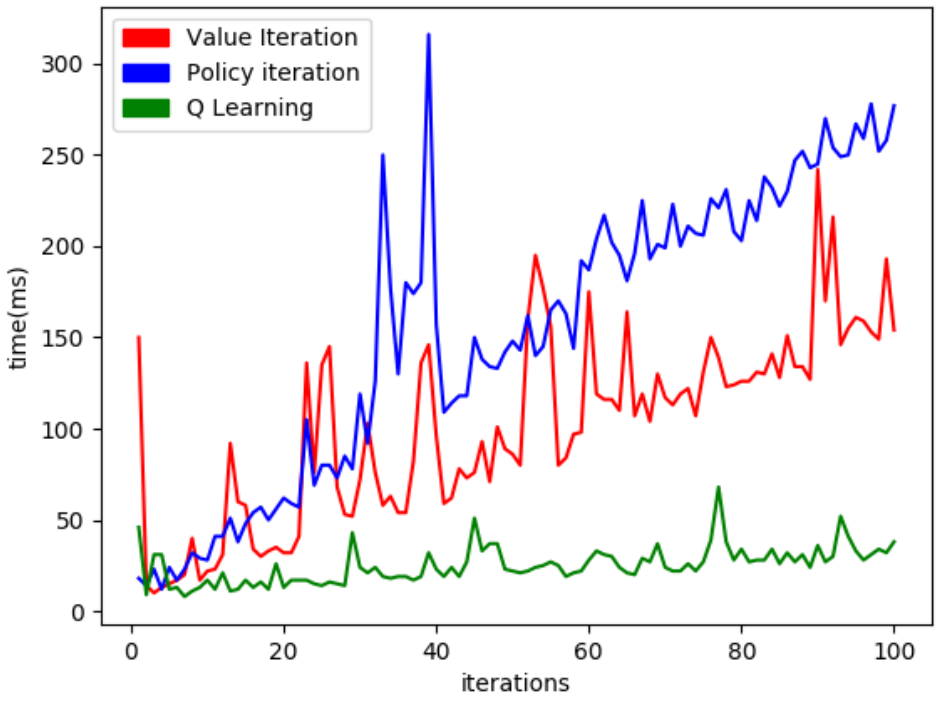


(Right: Reward/Iteration)(Left: Steps/Iteration)

Based off of these graphs we can see that it took the easy grid world about 5 iterations to converge for both value and policy. The hard grid world took about 10 steps for Policy iteration to converge, while still taking about 5 steps for value iteration to converge. It is worth noting that both of these graphs converge to the same reward

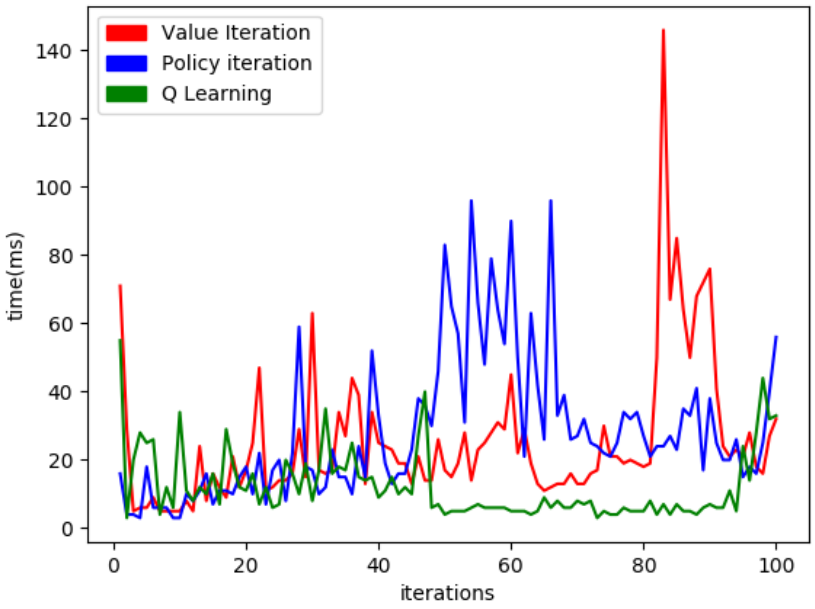
and number of steps. Value policy converges the fastest in terms of iterations on the HGW, while for the EGW they converge at the same number of iterations. In terms of computer runtime, we can generate the following analysis:

Hard Grid World Runtime Analysis



This clearly shows that value iteration is computationally much more efficient than the policy iteration on the HGW.

Easy Grid World Runtime Analysis



This shows that for the easy grid world, there isn't much computational advantage to using either policy iteration or value iteration.

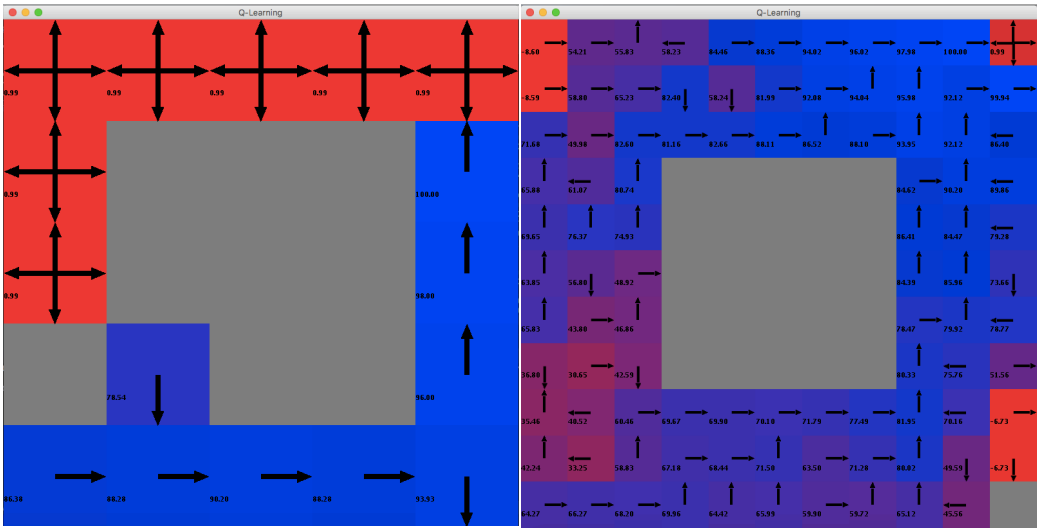
In the HGW, they end up generating the same policy, while in the small grid world they behave differently at the northern bound of the grid world. In terms of reward, they do converge to the same value.

Increasing the number of steps truly shows the performance of these algorithms, both in speed of convergence as well as computational runtime.

Q-Learning Algorithm

By no matter of happenstance, my favorite reinforcement learning algorithm happens to be Q-Learning. Q-Learning is a powerful algorithm that is able to find the optimal policy for any Finite MDP. We can see for the small grid world, Q-learning

generates a policy that is strikingly similar to that of policy iterations, albeit minor differences. We can typify Q-Learning best with HGW. The experiment was conducted with gamma = .99. qInit = .99, and learning Rate = .99.



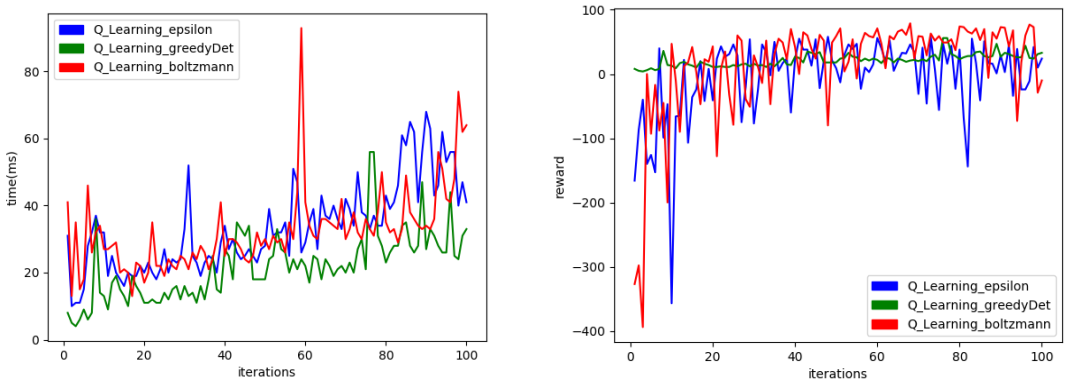
In the large grid world Q-learning generates a much more complex policy, than seen before with policy and value iteration.

Q-Learning is plotted alongside other algorithms earlier in the paper, and you can see that it converges to the same reward value as both Policy Iteration and Value Iteration, albeit with more variation around the value of convergence – makes sense when looking at the complexity of the policy produced. Remarkably, it can be seen that Q-Learning runs very fast compared to both Policy and Value Iterations.

It is worth noting that I'm unsure of which exploration policy was being executed in the aforementioned Q-learning algorithm trials, however a rough glance at the following graphs would note that it is nearly identical to Epsilon Greedy with a slightly

variant epsilon value. As such I've included a more thorough investigation of 3 different common Q-Learning exploration policies.

We run Epsilon Greedy exploration with an epsilon = .5, and Boltzmann Policy with .5.



From these graphs, we can see that Greedy Deterministic policy is the most consistent policy to use in regard to rewards per iteration. In terms of computation time, Greedy Deterministic also is the fastest. Both Epsilon Greedy, and Boltzmann policies produce significant oscillations around the general reward range. For the purposes of this Q-Learning task, Greedy Deterministic performs the best, or at least most stable and computationally efficient.