MARKOV DECISION PROCESSES

Aayush Kumar

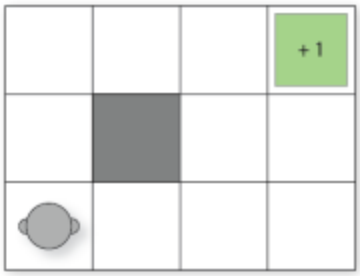
T-SQUARE ID: akumar372

Introduction

MARKOV DECISION PROBLEMS

In order to explore the effectiveness of Value Iteration, Policy Iteration, and Reinforcement Learning techniques on my two problems, I wanted to chose two problems of the same type but of varying levels of difficulty in regards to the number of states. Therefore, Grid World, although a frequently used problem for this type of study, did indeed lend itself to understanding the differences in the pros and cons of each algorithmic approach.

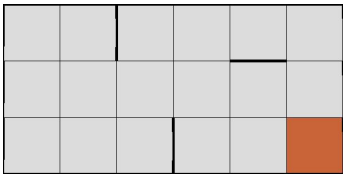
Grid World Problem Specification



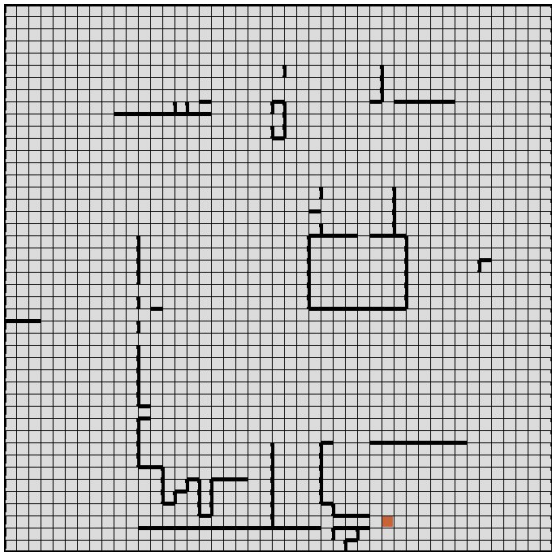
In grid world, an agent is trying to navigate through a maze of cells and walls in order to find a goal state- each cell indicates a possible discrete state for the agent to be in, and from there the agent has 4 stochastic actions it can attempt: move up one cell, move down one cell, move left one cell, move right once cell. However, these actions can occasionally be rejected due to the agent running into a wall either within the maze or bordering the maze. Optionally, the intended action can occasionally be replaced with an alternative action due to the stochasticity nature of the transition. In this problem, reinforcement is negative and thus any time MDP tells us to maximize reward, we are simply minimizing punishment.

More specifically, I designed my two grid world mazes such that there was a varying amount of uncertainty in each world where the intended action was only taken part of the time (PJOG) and otherwise one of the other remaining available actions would be uniformly chosen (1-PJOG/N-1). Apart from modeling noise in our environments, I held the threshold for convergence the same at 0.001 and in both cases and enforced a constant penalty of hitting a wall (Wall Penalty = 50). Both problems were designed as such in order to see how well each algorithm handled more and more uncertainty in the world as well as efficiency and other performance metrics outlined below:

6 x 3 Maze, 18 states



45 x 45 Maze, 2025 states



Algorithm Analysis

1. VALUE ITERATION

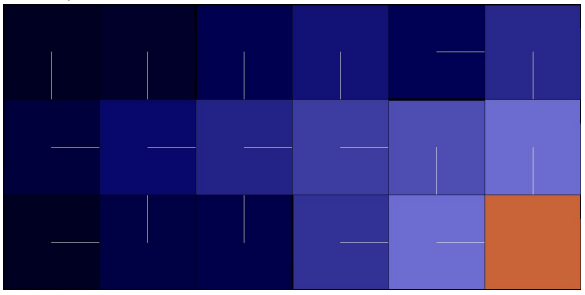
Description

We see that Value iteration is in essence an instance of dynamic programming in the computation of the values, which are stored and updated throughout the algorithm:

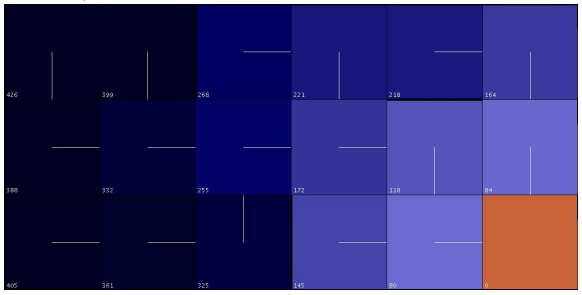
$$V_{t+1}(s) = \min_{a \in A} \left(PCost + \sum_{s'=S} (P(s'|s,a) \cdot x_{ss'}) \right)$$ where PCost is the path cost, P(s'|s,a) is the probability of a transition from s to s' via taking action a, and x_{ss'} is V_t(s') if the transition from s to s' is valid and otherwise is the defined Wall Penalty + V_t(s). Value iteration aims to calculate how helpful it is to be in a certain state in relation to the goal state, and the propagates this until we have computed a path that connects our start state to goal state that does not significantly change after running more updates (defined by our threshold of 0.001).

Small Grid World

44 steps, 4ms, PJOG = 0.3

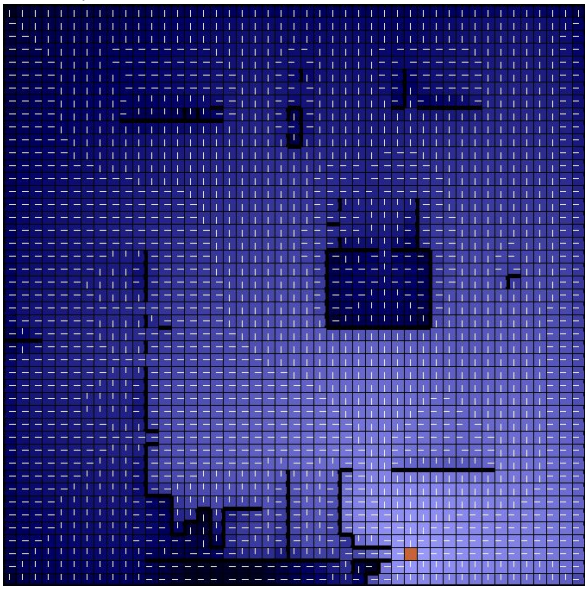


191 steps, 36ms, PJOG = 0.6

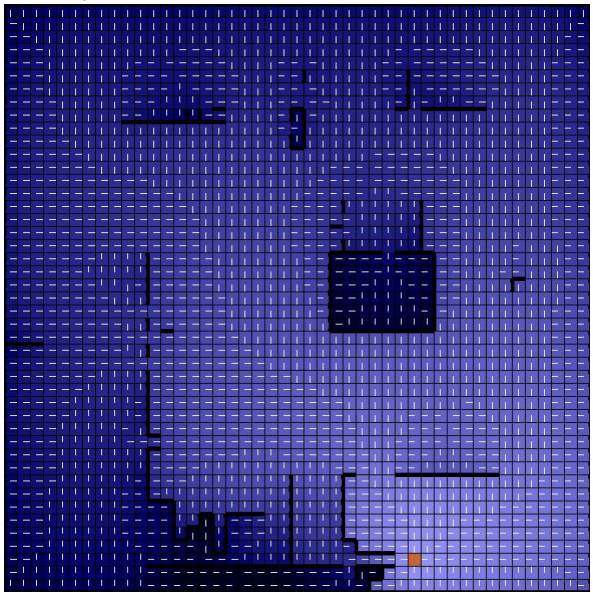


Large Grid World

181 steps, 2826ms, PJOG = 0.3



956 steps, 15114ms, PJOG = 0.6



Analysis

We can see that with more stochastic uncertainty in the world with increasing the PJOG parameter, both the time and number of iterations it takes to converges increases notable. There is an evident direct correlation here, and this can likely be explained by the notion that an agent has to make more decisions to account for the increased uncertainty of taking its intended action. Even if the agent has learned the correct step to take at state s, it should factor in the greater possibility that it will take an incorrect step and therefore much look into how to proceed from the unintended action in order to get back on track. This countermeasure propagates throughout every step taken in the Markov Decision Problem, and therefore we see that the resulting longer runtime increase by a larger and larger factor the more states a problem has. Similarly, we see that number of iterations it takes to converge to an acceptable answer also takes 5.28 times as much for the large grid world vs 4.34 times as much for the small grid world. We can guarantee almost certainly that this ratio will get larger as the grid world we consider gets bigger.

2. POLICY ITERATION

Description

Policy Iteration aims to estimate the local value of a state and use it to calculate how helpful taking a certain action is, thereby constantly improving the current policy directly until it derives the optimal policy. This two part process is outlined below, where we evaluate a first set of n linear equations with n unknowns:

$$V_{t+1}(s) = PCost + \sum_{s' \in S} (P(s'|s, \pi(s)) \cdot x_{ss'})$$

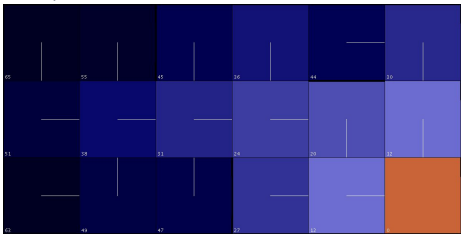
and then update the policy based on:

$$\pi_{t+1}(s) = \arg\min_{a \in A} (\sum_{s'=S} (P(s'|s, a) \cdot x_{ss'}))$$

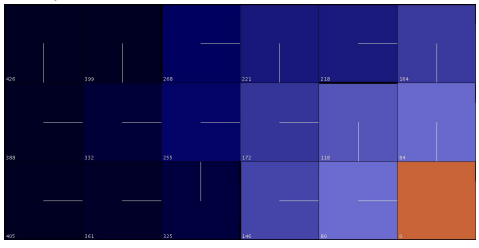
In practice, Policy Iteration is usually faster than value iteration, as by nature of the approach a policy converges more quickly than a value function.

Small Grid World

4 steps, 13ms, PJOG = 0.3

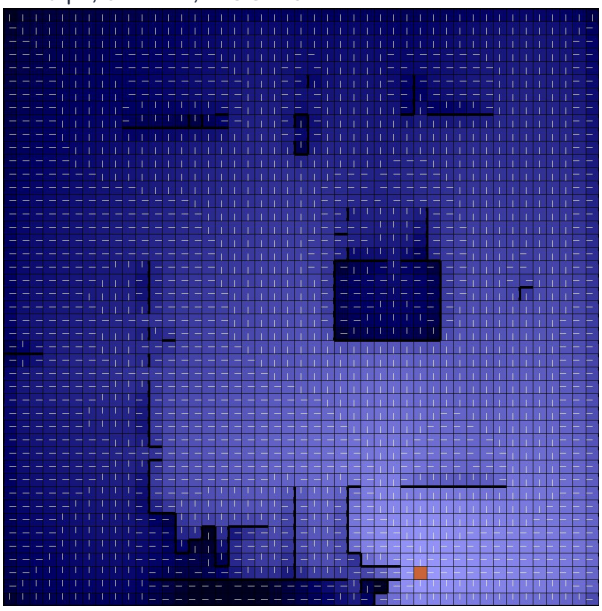


3 steps, 16ms, PJOG = 0.6

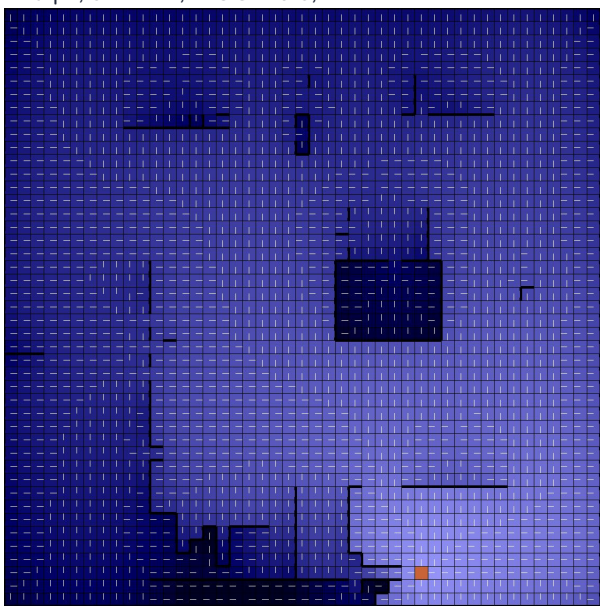


Large Grid World

15 steps, 6415ms, PJOG = 0.3



8 steps, 9773ms, PJOG = 0.6,



Analysis

As opposed to Value iteration, we see that policy iteration takes significantly less iterations to converge, but at the cost of each step taking much longer to calculate. We can attribute this to the increased computational complexity of taking the current step's policy and iteratively improving it until we converge to finally compute a value function. However, this also explains why it appears to handle stochastic noise much better, something we can observe in the notable smaller runtimes, as it can progress more rapidly as a result of the reduced number of possible policies as opposed to the possible number of values. As expected, we do also see a similarity to value iteration in that the larger the number of states as a result of the more complicated problem, the more time we will take to converge to a solution. However, we see that there are significantly less steps taken in order to solve the larger Grid World problem in relation to it's smaller counterpart than in value iteration. Specifically, it takes Policy Iteration about 2-3 times as many steps to derive a solution for the larger problem compared to the smaller one while Value Iteration takes about 4- 5 times. At the very least, this hints at the notion that for larger and larger problems with big state spaces, policy iteration is a superior candidate when it comes to solving the MDP.

3. REINFORCEMENT LEARNING

I chose to use Q Learning with Epsilon Greedy Exploration because it gives us an explicit parameter to understand the exploration vs exploitation tradeoff. More specifically, Epsilon Greedy Exploration gives use epsilon, a parameter that in many ways parallels the temperature parameter of simulated annealing. The larger epsilon is, the more we elect to explore other parts of our solution space while the smaller it is, the more we chose to believe what we learned at each step. The primary reason for not simply believing our incrementally learned policy is that perhaps we are stuck in a local optima, and similar to simulated annealing we need to jump out of that optima to explore other potentially even better solutions in hopes of targeting the global optima. Therefore, I decided to vary epsilon to specifically explore where an appropriate place to strike this tradeoff would be. (Learning rate was held constant at 0.6)

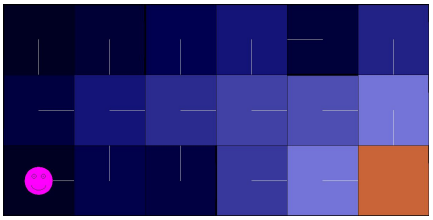
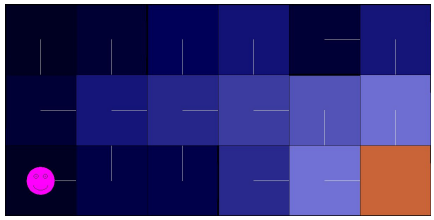
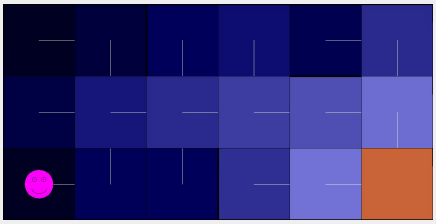
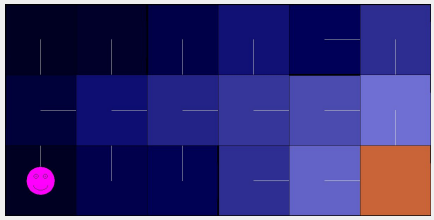
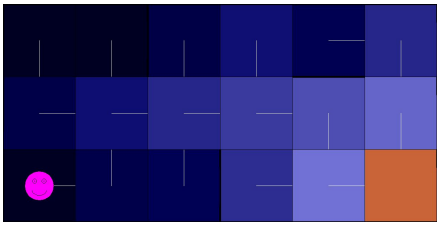
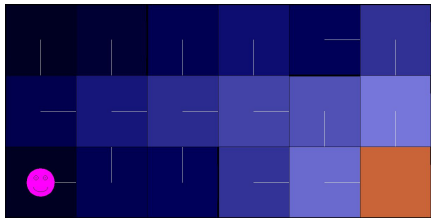
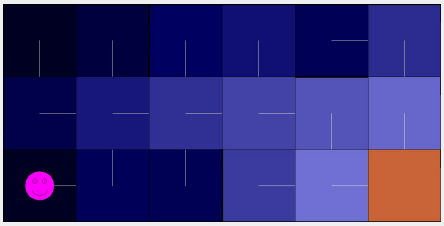
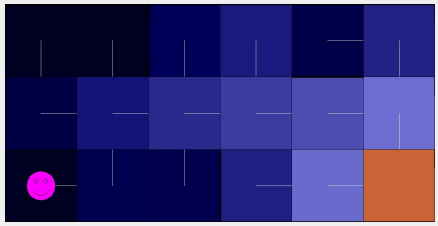
Perhaps the biggest difference in Q-Learning is that it does not assume a known model unlike how the algorithms seen thus far required a transition model and reward function. As a model-free approach, Q-Learning aims to learn the policy directly by scoring each state-action pairing, Q(s,a). After doing so for all possible pairings, the optimal policy is simply the best (in our case minimal) traversal of these pairings from the start state to the goal state. Q Values are obtained via initial guesses and received rewards:

$$Q(s,a) = Q(s,a) + \alpha \left(X + \min_{a' \in A} \left(Q(s',a') - Q(s,a) \right) \right)$$

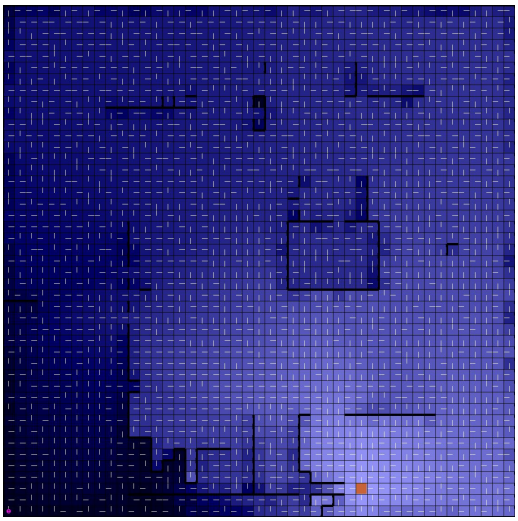
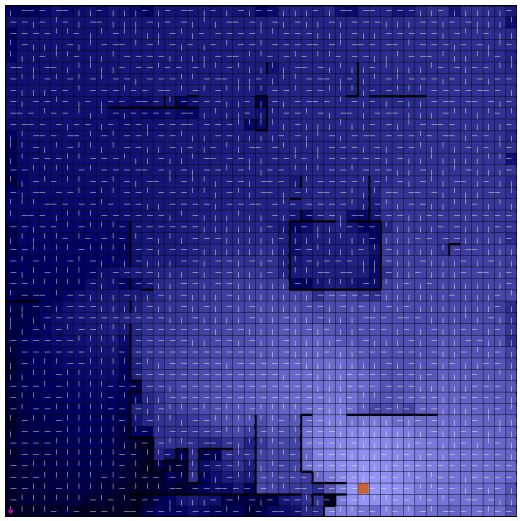
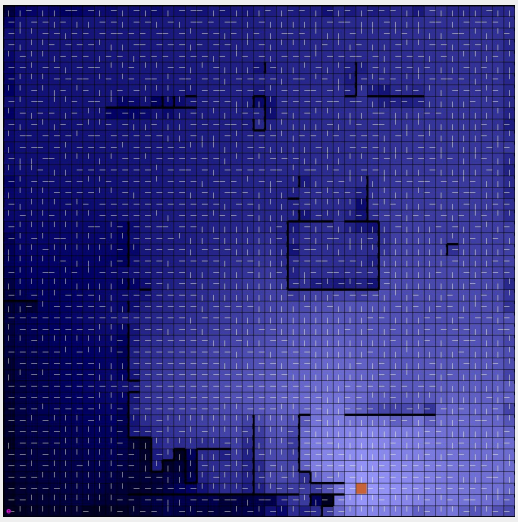
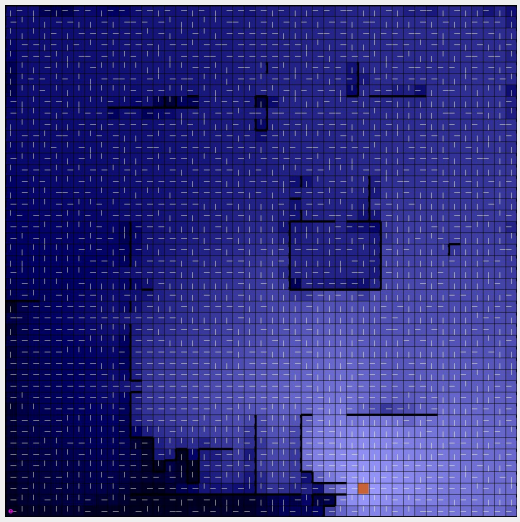
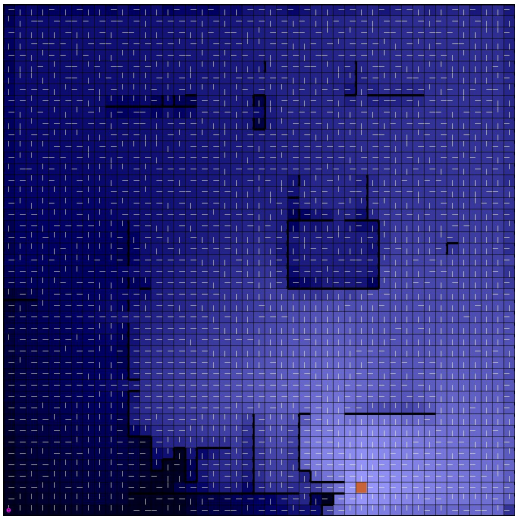
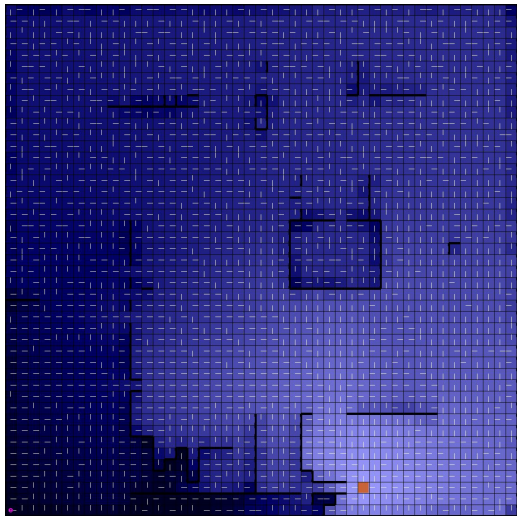
which is then traversed minimally to produce the optimum policy:

$$\pi^*(s) = \arg\min_{a \in A} \left(Q^*(s,a) \right)$$

Small Grid World Results

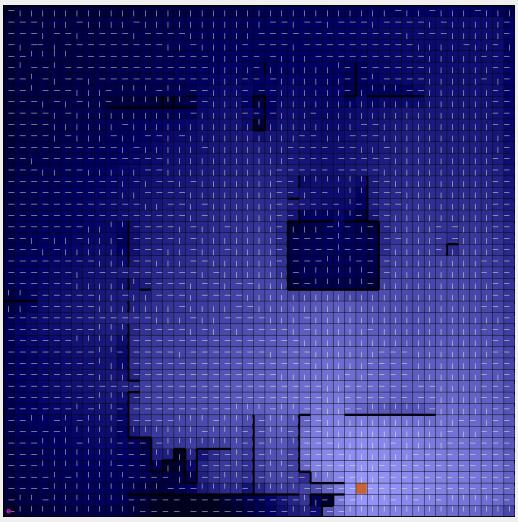
	PJOG = 0.3	PJOG = 0.6
epsilon = 0.1	744ms 	748ms 
epsilon = 0.3	1114ms 	1041ms 
epsilon = 0.5	1416ms 	1422ms 
epsilon = 0.9	4793ms 	5089ms 

Large Grid World Results

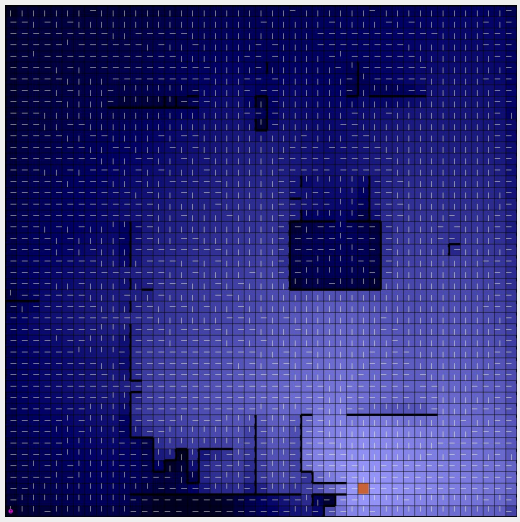
	PJOG = 0.3	PJOG = 0.6
epsilon = 0.1	10949ms 	10834ms 
epsilon = 0.3	15126ms 	12605ms 
epsilon = 0.5	18439ms 	20664ms 

epsilon = 0.9

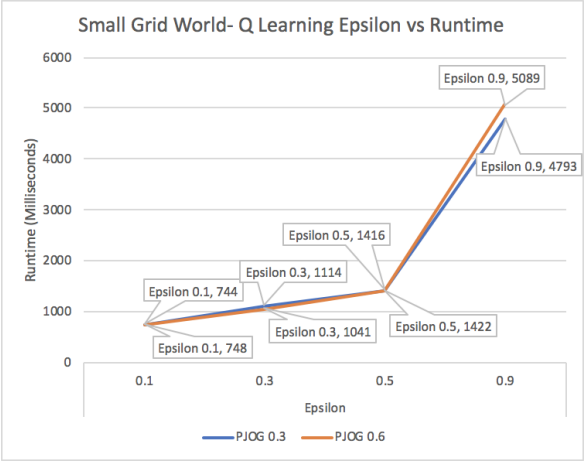
148772 ms

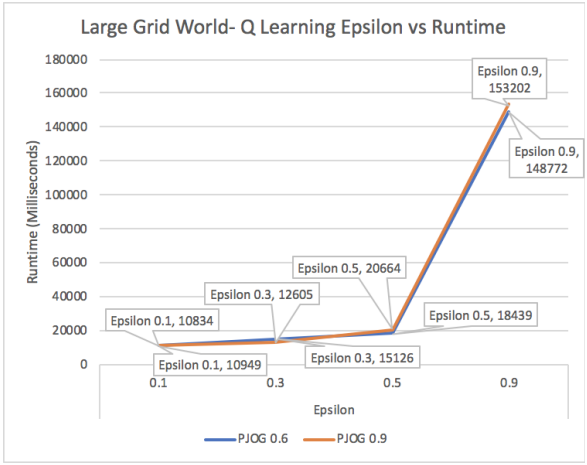


153202ms

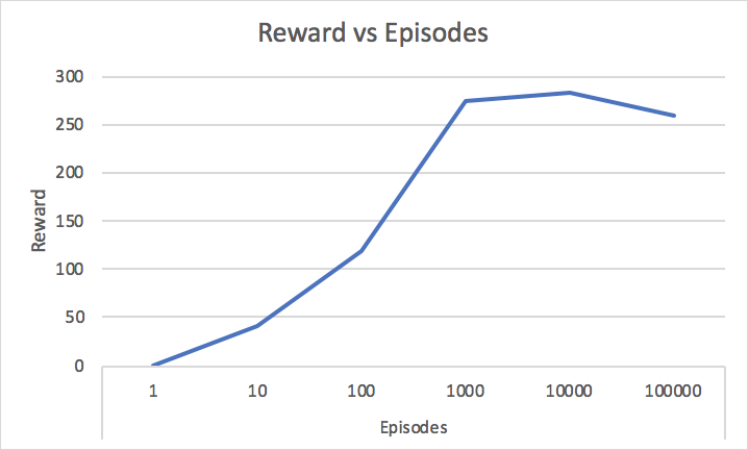


Analysis





After taking out all logging statements, we see that Q-learning takes lessf time to run when compared to both policy and value iteration. More formally, as the value of epsilon was increased, we chose to explore more often than exploid and the outputted policy was quite similar to the optimal policy generated by policy iteration. On the other hand, the more we chose to exploit by keeping a lower epsilon, the more difference we saw in the output policy when compared to the optimal policy generated by policy iteration. This is quite reasonable, as this lines up with the notion that exploration plays a more important role than exploitation for Q Learning to learn enough and improve the generated policy. Without exploration, we see most of the differences in the policy near the walls of the maze, which can be explained by the idea that a policy with less stochastic uncertainty is less likely to end up closer to the walls of a maze due to their negative penalty nature.



We also see that with more and more episodes, we are able to accumulate more reward (this is converted from our case of minimizing punishment). However, we do notice that after a certain point there are diminishing returns when it comes to adding more episodes to our Q-Learning run. Notably at the 10000 episode mark we see our best result at a score of 282.5.

Conclusion

When considering the results of our experiments in the context of Grid World, we see how each of these algorithms is able to achieve an ever so slightly different result with various tradeoffs. We saw that value iteration and policy iteration have some inherent similarities in how they are computed, but can say that policy iteration appeared to be the superior performer simply due to its ability to converge in fewer, albeit more expensive, iterations and scale better for larger problems as well as handle noise better. As expected, increasing stochastic uncertainty in the world as well as simply the number of states increased runtime exponentially. We also observed how Q-Learning benefits more from exploration vs exploitation in the classic tradeoff of Reinforcement Learning, particularly when the state space gets large. Even so, for smaller problems, we see that value iteration and policy iteration often outperform Q-Learning and this poses questions for how the three would perform with an entirely different Markov Decision Problem, perhaps one with continuous states and/or more actions.

Citations

1. https://classroom.udacity.com/courses/ud262/lessons/684808907/concepts/6512308840923
1. http://www.cs.cmu.edu/~awm/rlsim/
2. https://github.com/theJenix/rl_sim_fork