Jackson Cook

CS4641 – Machine Learning

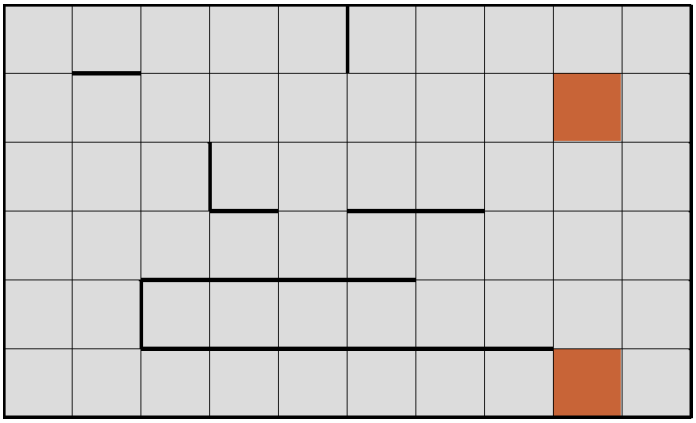
Assignment 4: Reinforcement Learning

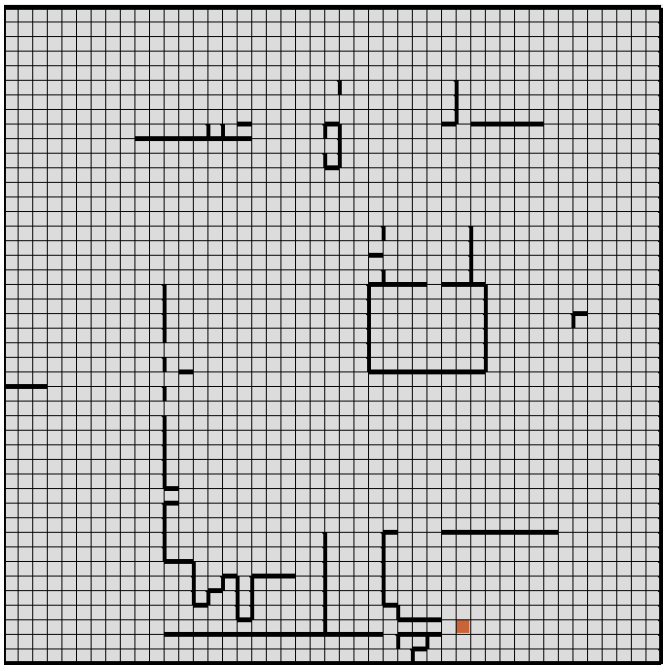
Markov Decision Processes (MDPs) are a method of modeling decision-making. In a discrete manner, the process evaluates what the best decision or "action" at every given time step or "state". MDPs are interesting because they allow you to model and optimize complex problems such as playing chess.

Both Value Iteration and Policy Iteration use the Bellman equation: $V(s) = \sum_{s'} T(s,s')(R_a(s,s') + yV(s'))$. V(s) is the value of the current state, T(s,s') is the transition value (the probability of actually going from state s to state s' if that action is chosen), R_a(s,s') is the reward of going from state s to state s', y is the decay constant, and V(s') is the value of the next state.

Part 1:

*Come up with **two** interesting MDPs. **Explain why they are interesting from an ML perspective.** They don't need to be overly complicated or directly grounded in a real situation, but it will be worthwhile if your MDPs are inspired by some process you are interested in or are familiar with. It's ok to keep it somewhat simple. For the purposes of this assignment, though, make sure one has a "small" number of states, and the other has a "large" number of states. **(10 points)***





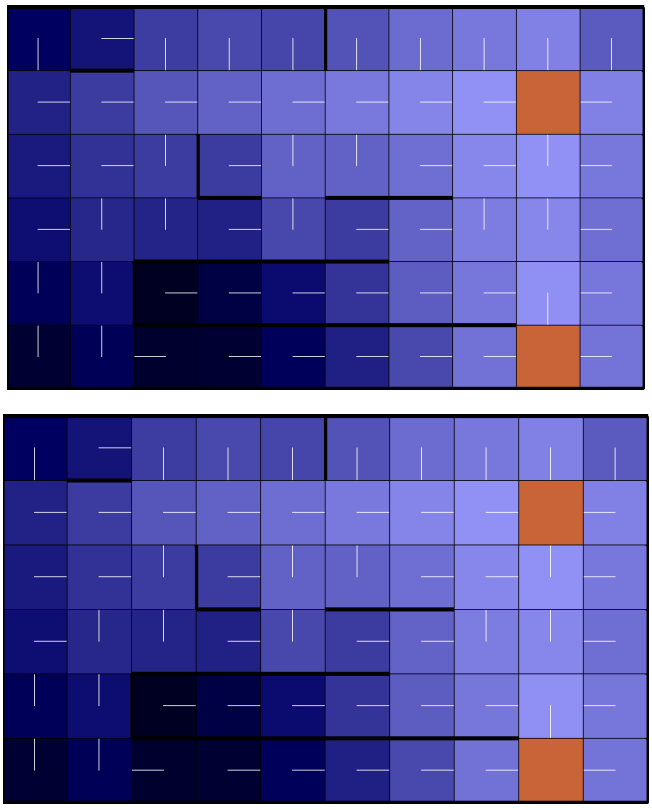
Images 1 and 2: Image 1 on the left shows the "Small Maze" which has 2 goal states while Image 2 on the right shows the "Big Maze" which has a single goal state and a significantly larger number of states

I have chosen to evaluate mazes to explore Markov Decision Processes. The maze above on the left (the "Small Maze" 10x6) has a smaller state space but has two goal states, giving the MDP a tough decision on which goal state to approach. The maze above on the right (the "Big Maze" 45x45) has a larger state space and only one goal state. I expect the Big Maze to take longer to converge due to its perceivably larger state space, but the large complexity introduced by the second goal state in the Small Maze might cause it to take more iterations to converge. The way this MDP works is by trying to minimize the penalty, losing 1 for every movement and losing 50 for hitting walls. Reaching the goal state just ends the penalty and does not yield a reward.

Part 2:

*Solve **each** MDP using **value iteration (15x2 points)** as well as **policy iteration (15x2 points)**. How many iterations does it take to converge? Which one converges faster? Why? Do they converge to the same answer? How did the number of states affect things, if at all?*

Small Maze



Images 3 & 4: Image 3 on the top is the result of running Value Iteration on the Small Maze while Image 4 on the bottom is the result of running Policy Iteration on the Small Maze

Both of these were ran with a transition value of 0.9. Value Iteration took 27 iterations to converge at a confidence of 0.0001 while Policy Iteration took only 5 iterations with the same confidence. This is expected, as Value Iteration is attempting to find the optimal function for the space while Policy Iteration is finding the optimal policy/choice at each state which is much simpler due to their only being four choices at each state; therefore, policy iteration should converge faster. Both methods converged to the same answer due to both of the methods using the Bellman Equation in their calculations. As can be seen on the bottom row, the explorer attempts to avoid going through the tunnel since it is penalized if it hits the wall.

Big Maze

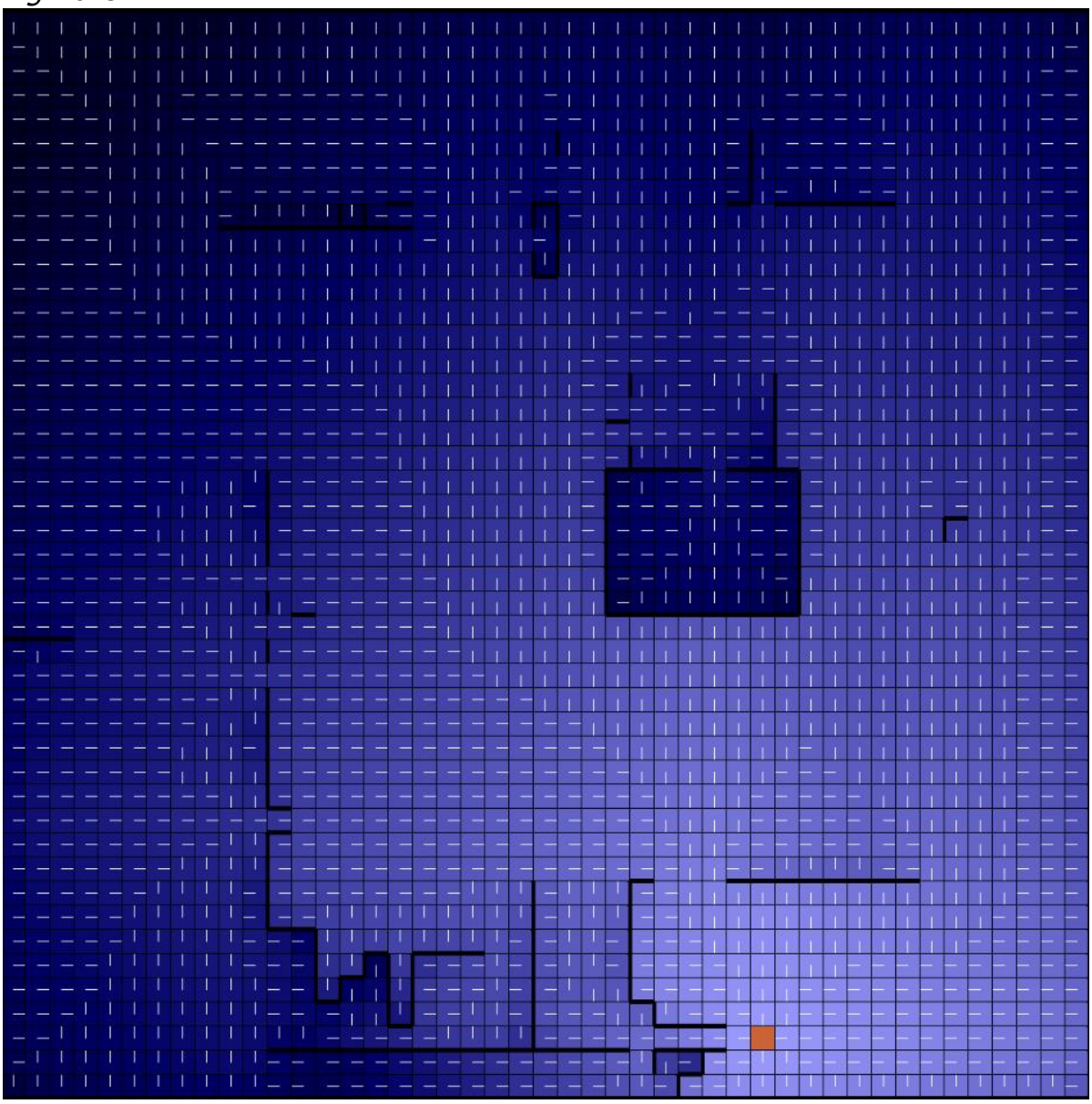
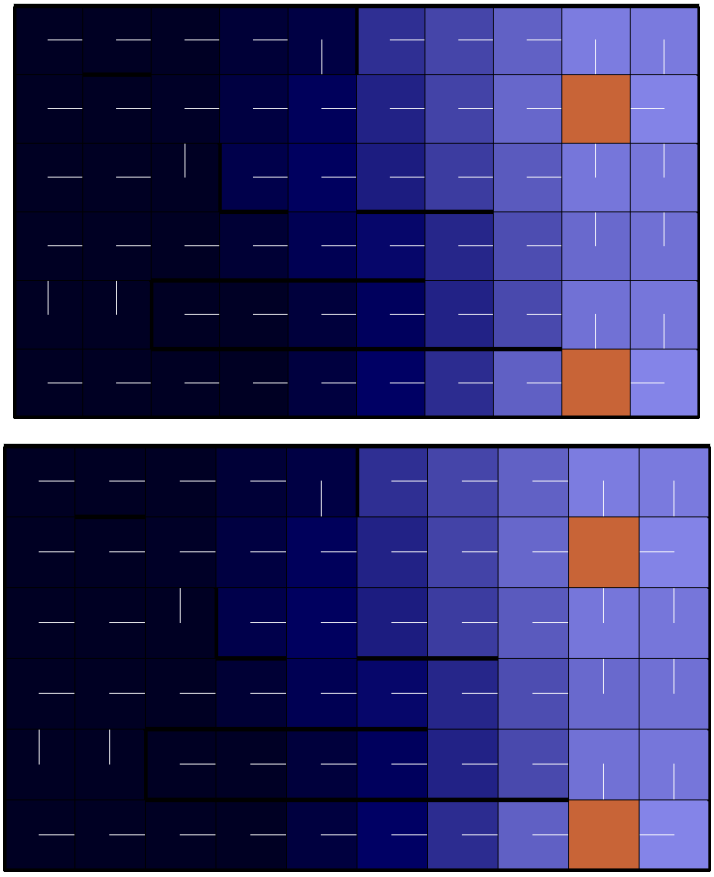


Image 5: Image 5 is the result of running Value Iteration and Policy Iteration on the Big Maze, two pictures were not included because it would be redundant and the picture is already so large

These were ran with the same rewards, transition value, etc. Value Iteration took 106 iterations to converge with a confidence of 0.001 while Policy Iteration took 49 steps to converge to the same result. Once again it is probably due to how Value Iteration searches for the optimal function while Policy Iteration searches for the optimal decision at each state.

Part 2 Conclusion

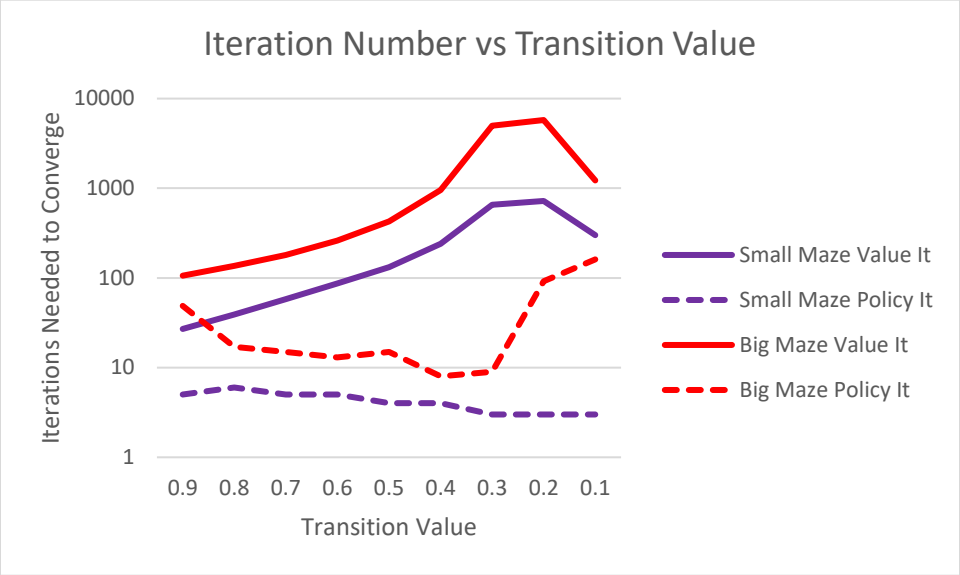
The Big Maze (the maze with the larger state space) took noticeably longer to converge for each method as well as taking about 4 times as many iterations for Value Iteration and nearly 10 times as many iterations for Policy Iteration. Making optimal decisions in a large space takes more optimization than it does in a small space as there is more to consider for the learner.



Images 6 & 7: Rerunning value iteration w/ transitions of 0.26 (top) and 0.24 (bottom)

I also wanted to look at how once the threshold of 0.25 was crossed for the transition value what happened. In this test, the transition value says the likelihood you are able to move in the direction that you want in the maze. The remaining probability (.75) is distributed evenly among the other 3 directions. So as you can see in the images above, when the transition value is 0.24 the policy is no longer to go into

the goal state when directly adjacent to it but rather to go away from it, as the other direction has a probability of reaching the goal of 0.25.



Graph 1: Shows number of iterations to converge based on a changing transition value

As expected, Policy Iteration was nearly always fewer iterations than Value Iteration. It can be noted that at a transition value of 0.1, the Value Iteration performs better on both the Small Maze and the Large Maze than at a transition value of 0.2, breaking the trend of constantly increasing iterations with decreasing transition value. As mentioned previously, the MDP chooses to not move in the direction it wishes to go after a transition value of 0.25. Since the transition value is 0.1, that direction that the MDP does not want to go is simply less likely than going that direction at a transition value of 0.2, and going the direction you want to go is more likely (0.3 vs 0.2666). The iteration number for the Big Maze with Policy Iteration changes in ways very different to the other three curves, but the reason is unknown to me.

I did analysis of the number of iterations needed to converge with changing confidence but find it unnecessary to include a graph, for as expected the number of iterations needed to converge decreases as the confidence decreases (needing leess iterations to reach a worse answer.

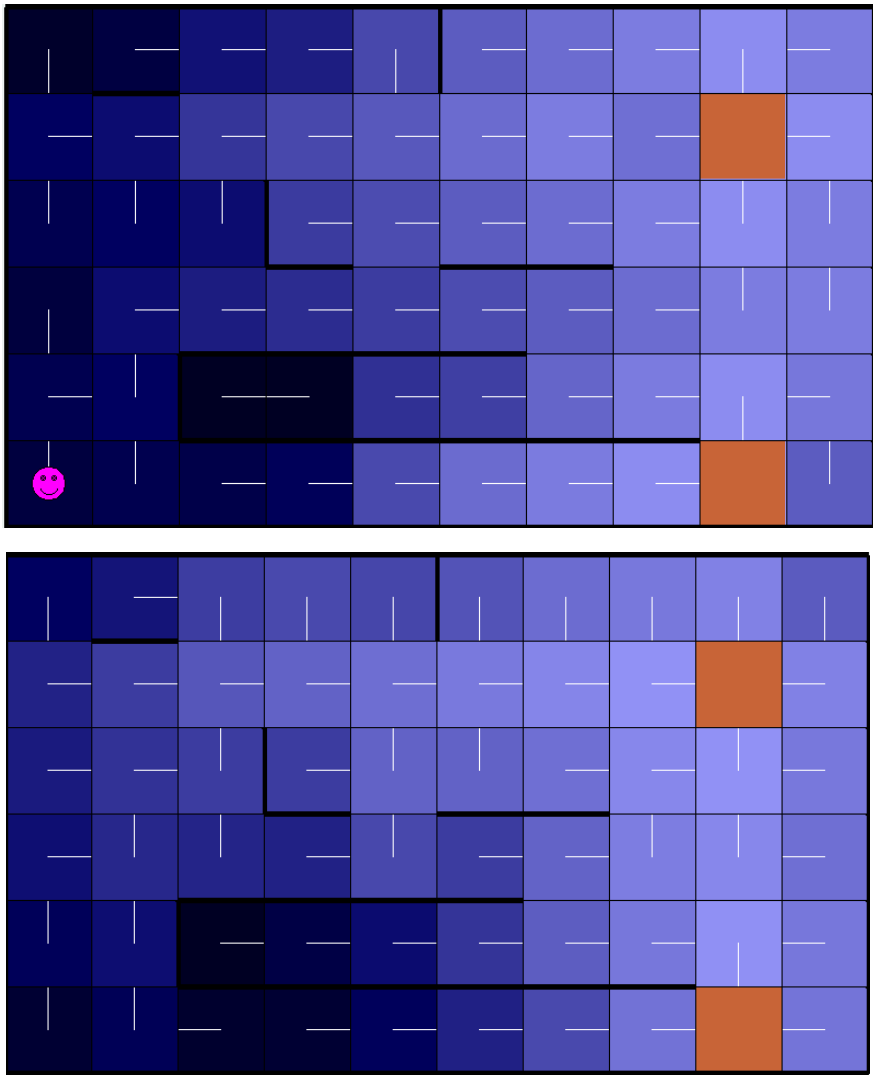
Part 3:

*Now **pick your favorite reinforcement learning algorithm** and use it to solve the two MDPs. How does it perform, especially in comparison to the cases above where you knew the model, rewards, and so on? What exploration strategies did you choose? Did some work better than others?*

I chose Q Learning to solve the two MDPs. The way I explored the space was by putting the explorer in the bottom spot of each maze and forcing it to randomly search

for the goal with decaying reward for each move. Then with a changing learning rate and epsilon (chance of randomly going a new direction) allowed the explorer to explore more. The strategy of starting in the bottom left corner every time unfortunately made exploring areas to the right of the goal states on both MDPs very difficult.

Small Maze



Images 8 & 9: Image 8 on top shows a run with high learning rate, low iterations, and medium epsilon while Image 9 below once again shows the results of Value Iteration on the Small Maze

I found that if I was trying to get some sort of convergence at a low number of iterations (100), it would converge the fastest with a high learning rate (0.99) and a high epsilon (0.8). This is because the epsilon would allow to ample exploration of the space while the high learning rate would cause the process to quickly pick up the new pathway. You can see that even with a pretty high epsilon that the right side of the maze especially to the right of the goal states is fairly unexplored. The process is able to make it to the goal state, and it is taking routes not that far from the optimal. When

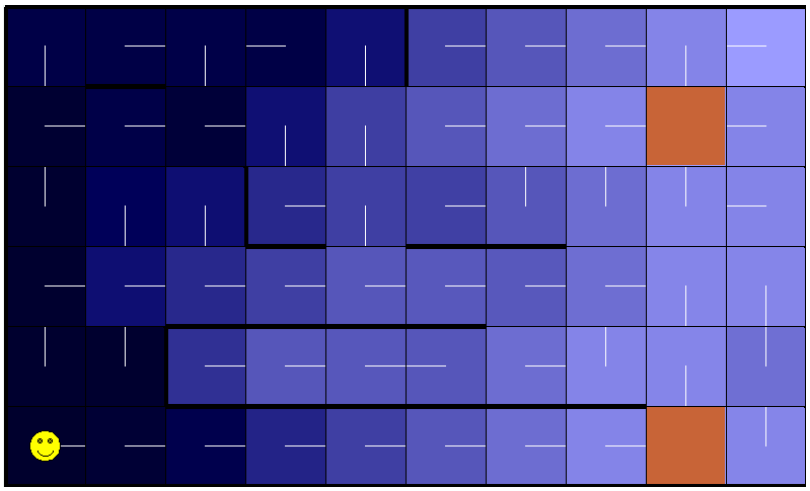
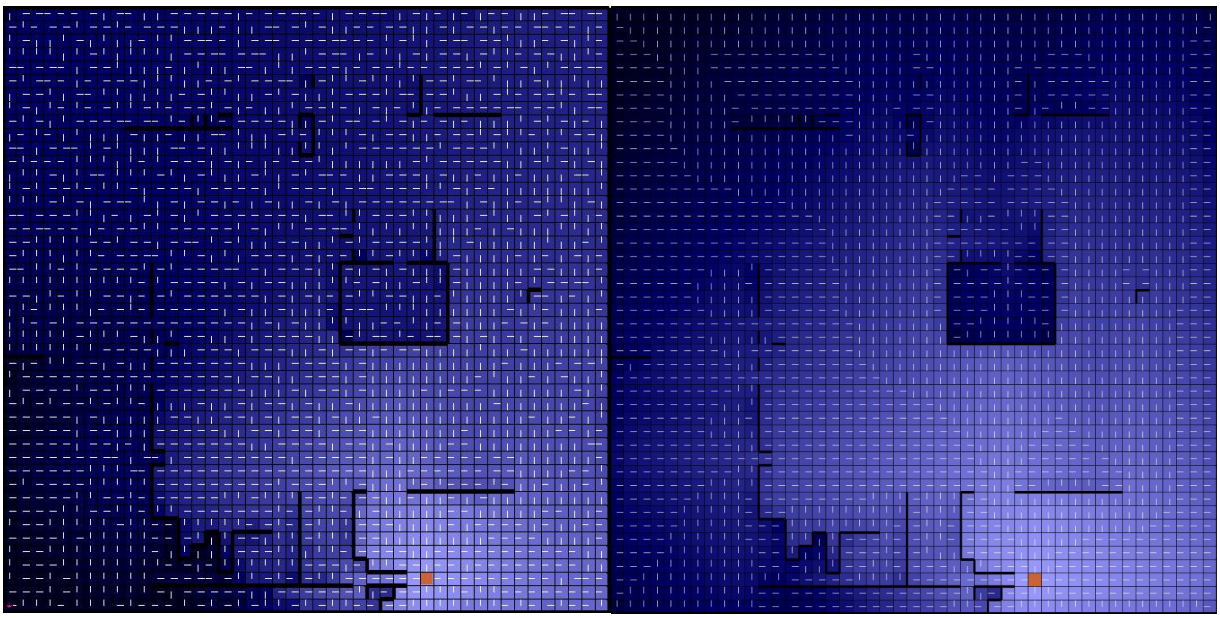


Image 10: Shows the results of running with the same previous parameters but fewer iterations (27)

I ran the Q Learner with the same parameters but fewer iterations (27, the number of iterations it took Value Iteration to converge), the results were much less accurate. This is due to lack of adequate exploration of the space despite the high epsilon (random exploration rate).

Big Maze



Images 11 & 12: Image 11 shows the work of the Q learner with epsilon 0.5, learning rate 0.8, and 10,000 iterations while image 12 shows the work of Policy Iteration on the Big Maze

Even with the max iterations of 10,000 I could run on my computer, the Q-Learner performed very badly on the Big Maze. Due to its large state space I would expect it to need many more iterations to even slightly resemble the policy of Policy Iteration. I tried many different epsilons and learning rates, but I was unable to see

any improvement from the learner. You can see that closer to the goal state and the walls, some of the choices are the same as those of the Policy Iteration policy, but the open spaces far from walls and goal state seem to be basically randomly chosen directions. Once again this is just due to the size of the space and the lack of iterations. When I ran the Q Learner with the amount of iterations it took for the Value Iteration to converge (106), There did not seem to be any patterns showing up due to the small iterations.

Conclusion:

To conclude, Policy Iteration and Value Iteration both converge to the same policy in the end, but Policy Iteration converges quicker and in fewer iterations. Q Learning takes far more iterations than Policy Iteration and Value Iteration and the number of iterations it takes for Q Learning to be effective increases much quicker with the size of the space than the number of iterations it takes Value Iteration and Policy Iteration to converge with the size of the space.