Ashwini Iyer
CS 4641
April 11, 2018

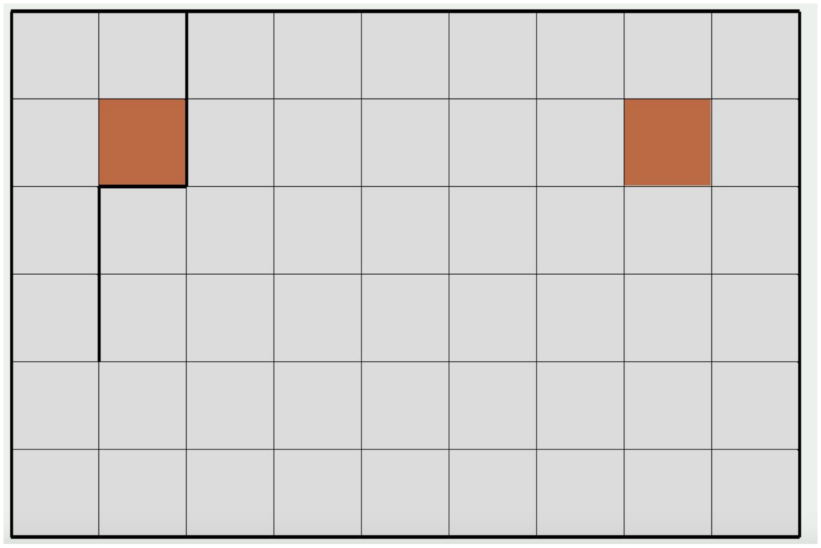
Markov Decision Models and Reinforcement Learning Report

Markov Decision Processes (MDPs) provide a mathematical framework for modelling decision making. They are particularly effective in situations where outcomes are a combination of randomized outcomes and decisions made by the agent. In this project we were asked to come up with two MDPs, one with small number of states and the other with a larger number of states. To model these problems I used the Grid World MDP, in which the agent can be in any one of the discrete physical states and can choose to move up, down, left or right. In order to encourage the agent to reach the goal state every action stated previously has a path cost. There is no reward for reaching the goal state, but there is also no path cost for it either. Hitting walls, on the other hand, incurs a penalty. Below are the MDPs I chose for this assignment.

MDPs Chosen

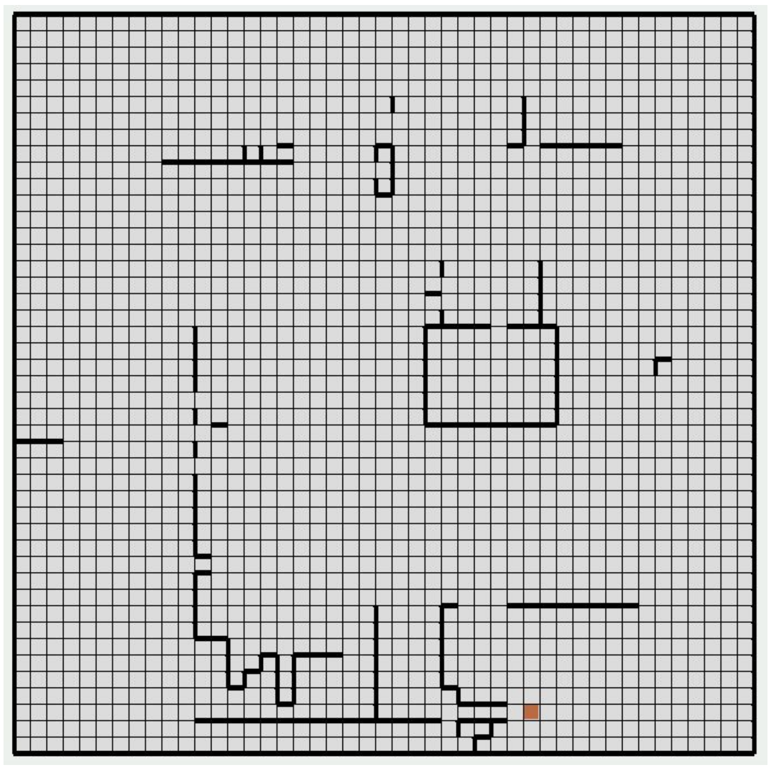
These MDPs are interesting because they will allow us to see how policy iteration and value iteration perform on different state sizes. The orange squares represent goals and the dark black lines represent walls. I chose these because I think they are very interesting to help model real life situations, such as helping robots navigate in the home or navigation tasks in general.

<u>Maze 1 (4_multipleGoals2.maze): 9x6 wall penalty 50</u>



This is a small grid that I used to represent an MDP with a small number of states. It has a 9x6 states and a wall penalty of 50. Which means that if the agent hits a wall it will get deducted 50 points. Notice how the goal on the left is covered by some walls and requires a particular navigation strategy to reach while the goal on the right is more open.

<u>Maze 2 (8_big.maze): 45x45 wall penalty 50</u>



This is a bigger grid that I used to represent an MDP with a large number of states. It has a 45x45 states and a wall penalty of 50. Which means that if the agent hits a wall it will get deducted 50 points. This bigger grid can be utilized to show a couple of key points. The agent is expected to realize that the enclosed square near the middle is a zone to be avoided and the agents behavior along the walls to the fastest paths to the goal state can be observed as well.

Value Iteration vs. Policy Iteration
A policy is a probabilistic mapping between actions and states, whereas values in reinforcement learning are usually for state estimation techniques. Value Iteration is an algorithm that works by trying to estimate the value of being in a specific state in the maze. You start with a random value function and then iterate to find new random value functions and eventually reach an optimal value function. Policy Iteration is an algorithm where we estimate the value of every state which is determined by their neighboring states. The two algorithms differ because in

Policy Iteration you directly change the policy of what actions you choose in each state whereas with Value Iteration you estimate the optimal value of a state and iteratively update the estimated state values. The deterministic policy selected for value iteration is extracted from the optimal value function where the agent moves towards the best state.

In my experiments that I ran using these two mazes, I chose to run policy and value iterations with the following parameters. For policy iteration I stopped evaluating the current policy if a value of any of the states exceeded 5000 and only evaluated the policy up to 500 iterations. For both value and policy iteration, I ran it with a PJOG of 0, 0.1, 0.3 and 0.9. PJOG represents in the noise or error in the environment. A PJOG of 0 and 0.1 represents a fairly deterministic environment (0 more so than 0.1), because all actions taken by the agent are effective 90% to 100% of the time. A PJOG of 0.3 indicates that 70% of the time an action produces the intended next state, and 30% of the time the action causes the agent to get pushed to any of the other states and not the one that was specified. A PJOG of 0.3 and 0.9 represents a stochastic environment. I was curious to see how the policies performed in these two different situations. I also used a precision of 0.001 for checking whether the algorithm converged. I also used a delay of 0, which represents the microsecond delay between the two successive steps in the animation of the tool I used to model these states.

Results

Below are the results gathered when Value Iteration and Policy Iteration were run on the MDPs. The only parameter changed was the PJOG. The first table includes the initial results where the PJOG is 0. The second table includes the results for a PJOG of 0.1 and of 0.3.

Maze	Algorithm	PJOG	Total Time (ms)	Iterations Needed
1	Value Iteration	0	2	9
1	Policy Iteration	0	8	17
2	Value Iteration	0	1553	74
2	Policy Iteration	0	11988	66

Maze	Algorithm	PJOG	Total Time (ms)	Iterations Needed
1	Value Iteration	0.1	8	24
1	Policy Iteration	0.1	14	5

1	Value Iteration	0.3	12	48
1	Policy Iteration	0.3	17	4
2	Value Iteration	0.1	2028	106
2	Policy Iteration	0.1	10,078	49
2	Value Iteration	0.3	3703	181
2	Policy Iteration	0.3	10,296	15

It is clear from the numbers that Value Iteration took much less time to converge on both mazes than Policy Iteration regardless of the change in PJOG. This is because Policy iteration needs to evaluate the entire Markov Chain from any start state resulting in an O(n^3) time complexity of states per iteration. Value Iteration results in an O(n) time complexity of states per iteration because it only needs to compute the values of the neighboring states.

However, Value Iteration required consistently more iterations to converge. This is a result of how the two algorithms terminate. Value iteration continues even though the agent's actions in a given state are no longer changing because it terminates only when no values change. Policy Iteration, on the other hand, terminates when the policy no longer changes. Therefore, Policy Iteration is more time efficient but takes orders of magnitude longer to execute per iteration. The difference in time taken and iterations needed is magnified when the state representation of the maze changed to the larger maze.

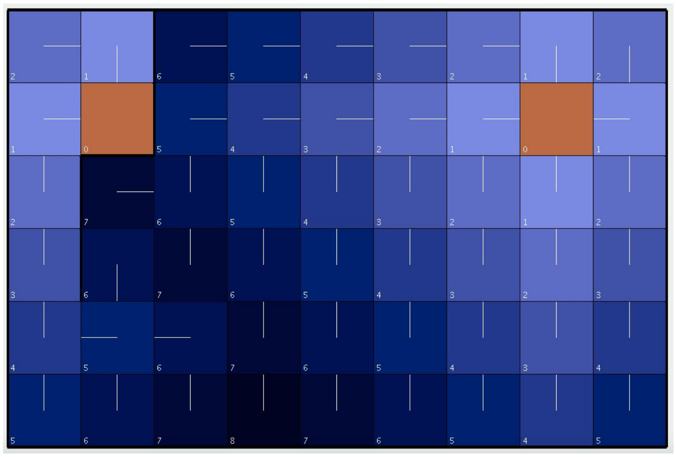
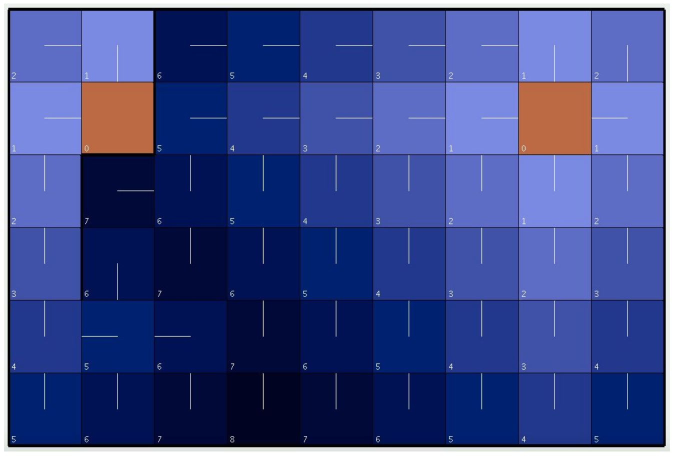
It is interesting that when the PJOG increased to 0.3 both maze 1 and maze 2 took both more time and iterations to converge with Value Iteration, but this makes sense because it is a much less deterministic environment. With a less deterministic environment, the noise in the reward signal takes longer to distribute to the states. But, with Policy Iteration maze 1 took roughly the same amount of time and iterations to converge regardless of PJOG. On the other hand, maze 2 took much less iterations to converge, but roughly the same amount of time with Policy Iteration. Therefore, it appears that Policy Iteration is more robust and resistant to changes in convergence with increased stochasticity in the environment possibly due to the entire Markov Chain calculation made during every step.

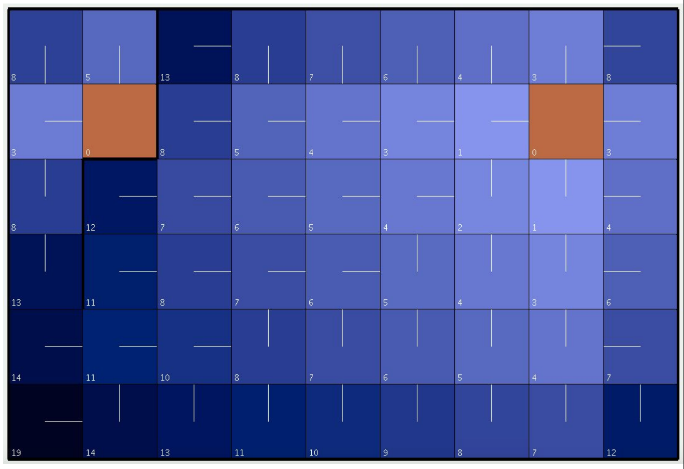
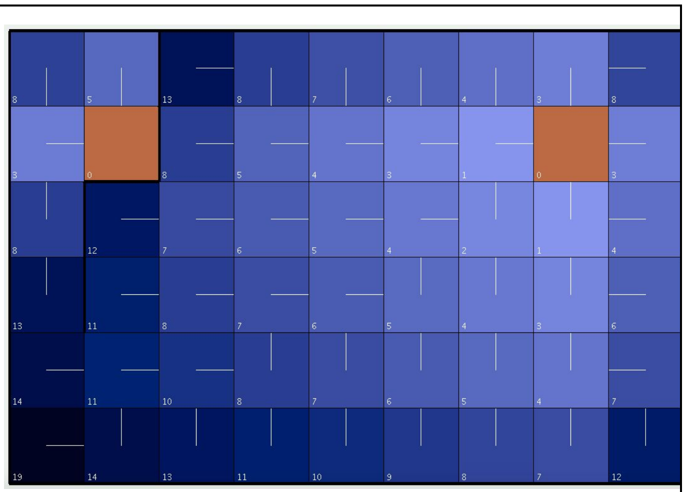
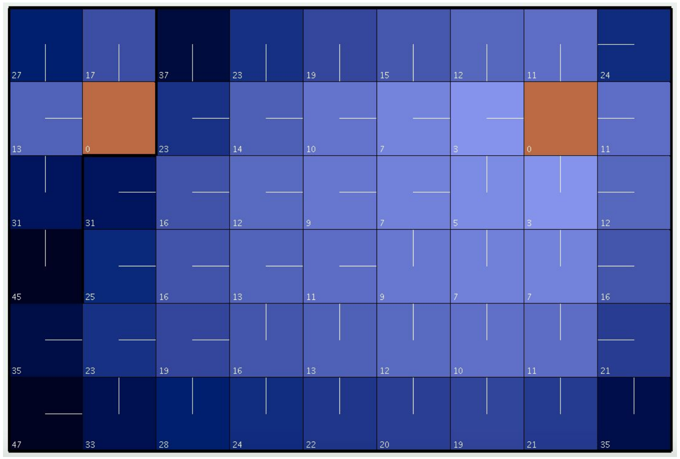
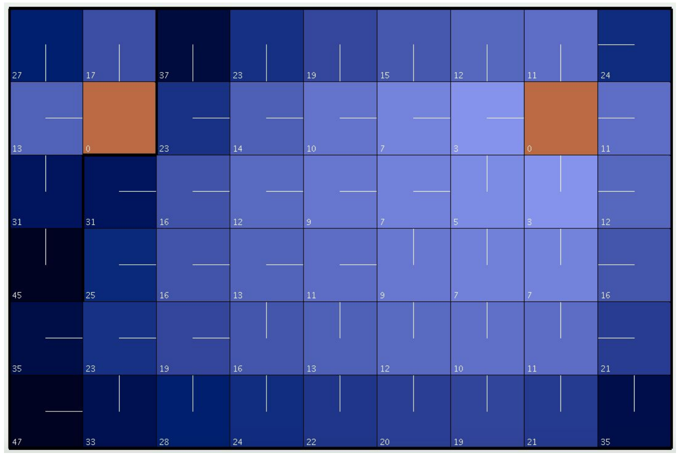
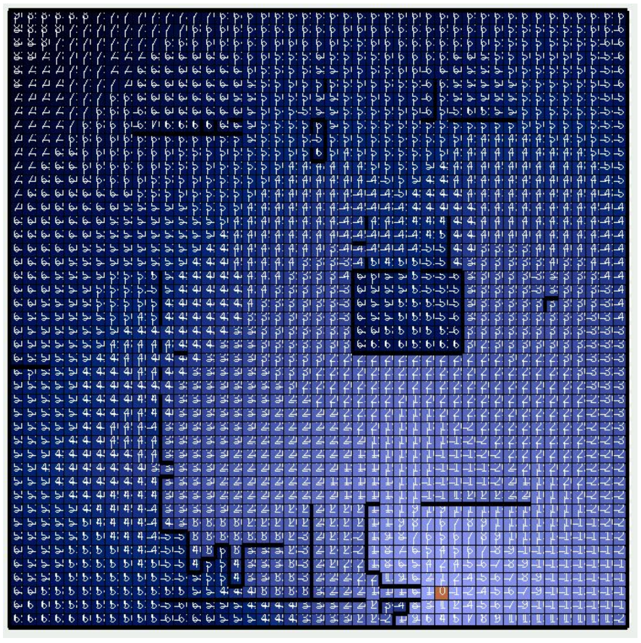
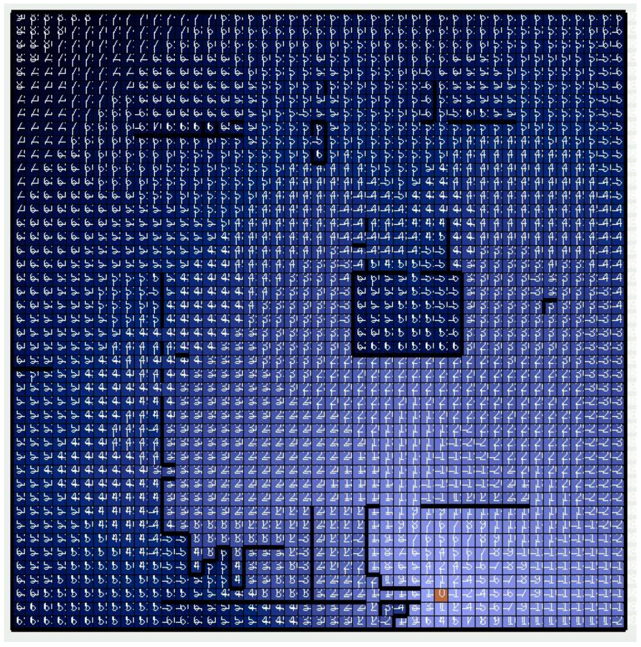
Increasing the number of states did increase the amount of time and iterations needed to converge. This is because the more states there are the more iterations needed to get the correct values for every state. In particular, for Policy Iteration this means that it takes longer to calculate the Markov Chain from each state and for Value Iteration it means that it takes longer to converge on the utility values for each state since there are more states. Because Policy Iteration scales with O(N^3), as the state space increases the time taken increases cubically.

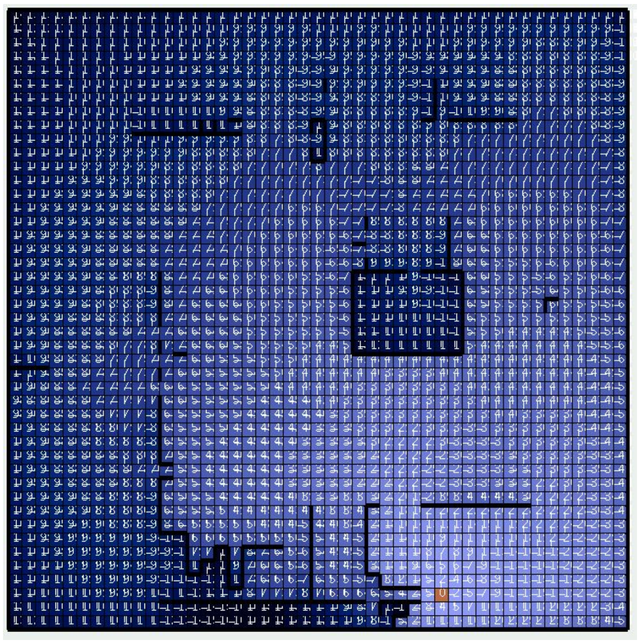
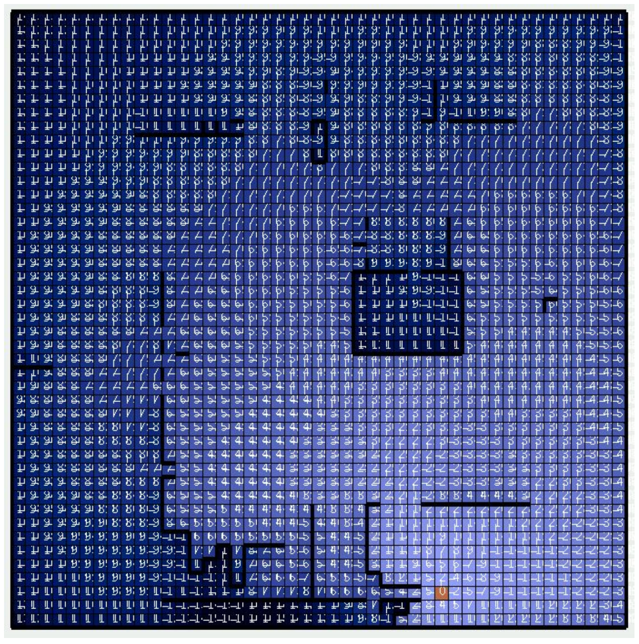
Below is a table with the final answers for Value Iteration and Policy Iteration for both mazes with both PJOG values. The lines in each box represent the direction the agent should go in that state. The lines representing walls are a bit faint in these pictures due to the shading of the boxes, please refer to the pictures in the beginning of the document that have a clearer depictions of the wall boundaries. Here we can see that for all cases the results for Policy and Value iteration are nearly identical. Slight variations may be accounted by differing converging policies.

Note how the policy (action chosen) in the states near the left goal changes as the PJOG value increases. When the PJOG was 0.0, the final policy tended to involve paths that were closer to walls because the agent was more willing to get close to the walls since it has a very small likelihood of accidently hitting it. When the PJOG was higher, the risk was too high for the agent to go near the walls. As such, the agent would stay clear of the walls and many of the actions along the wall were away from the walls even if the goal was closer with a parallel path along the walls. Deterministic environments, represented by a PJOG of 0.0, are easier to solve because the agent can count on their decisions being executed with little to no uncertainty.

In the below images the higher the score of the state and the darker the color of the square. As you can see around the goal the squares are a lighter blue, around the walls they are a darker blue.

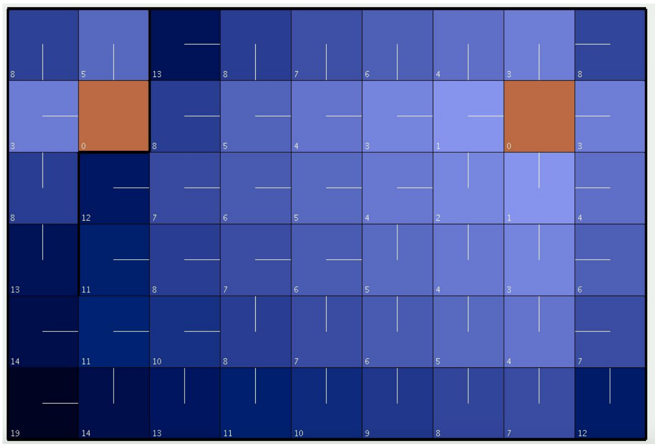
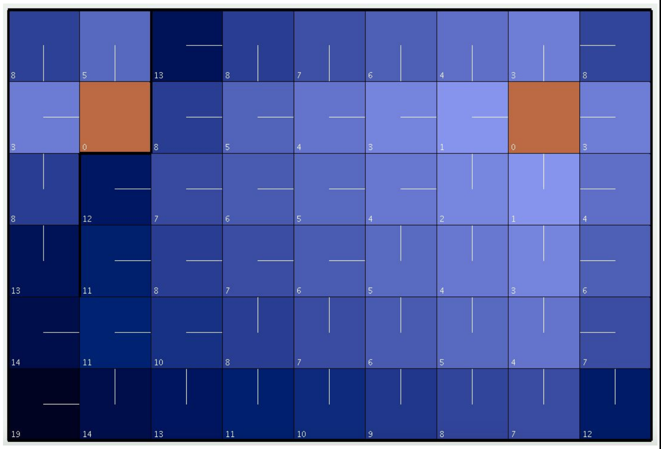
Maze, PJOG	Value Iteration	Policy Iteration
1, 0.0		

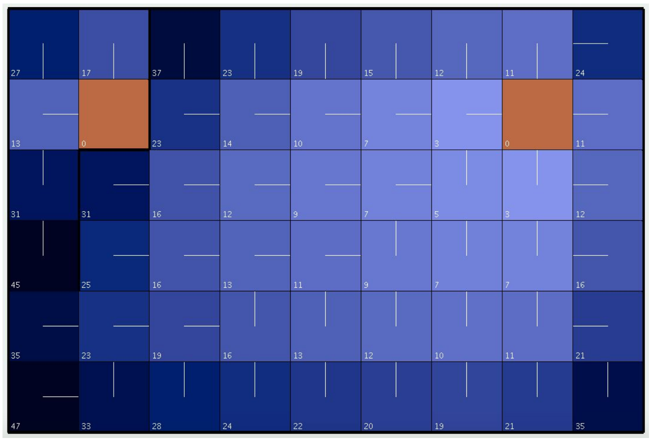
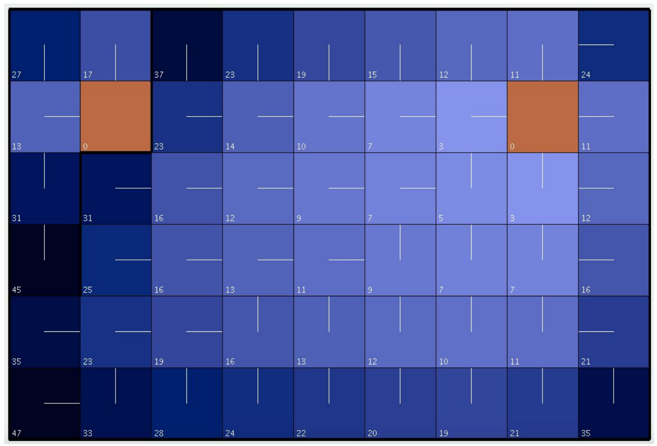
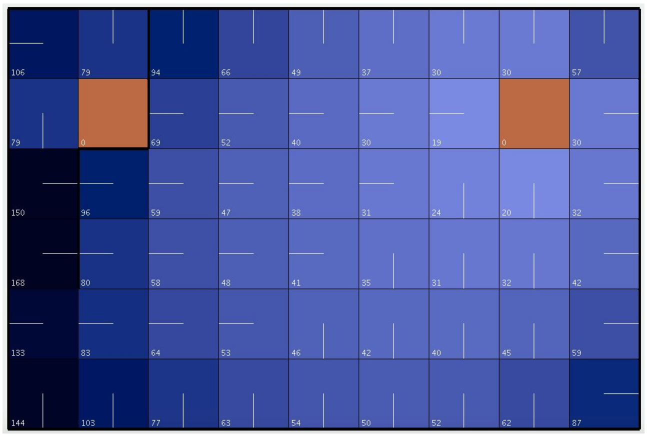
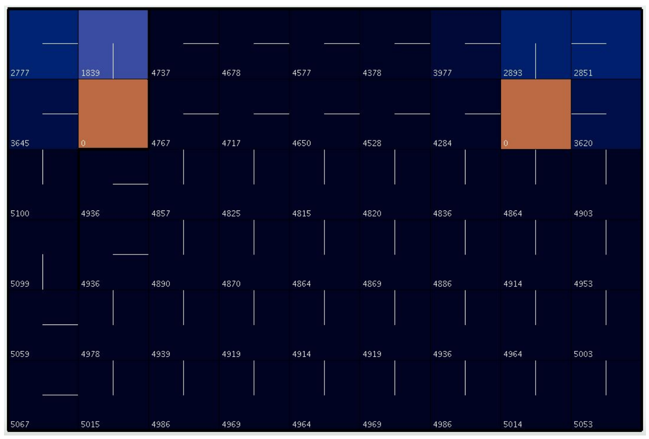
1, 0.1	8 5 13 8 7 6 4 3 8 3 0 8 5 4 3 1 0 3 8 12 7 6 5 4 2 1 4 13 11 8 7 6 5 4 3 6 14 11 10 8 7 6 5 4 7 19 14 13 11 10 9 8 7 12	8 5 13 8 7 6 4 3 8 3 0 8 5 4 3 1 0 3 8 12 7 6 5 4 2 1 4 13 11 8 7 6 5 4 3 6 14 11 10 8 7 6 5 4 7 19 14 13 11 10 9 8 7 12
1, 0.3	27 17 37 23 19 15 12 11 24 13 0 23 14 10 7 3 0 11 31 31 16 12 9 7 5 3 12 45 25 16 13 11 9 7 7 16 35 23 19 16 13 12 10 11 21 47 33 28 24 22 20 19 21 35	27 17 37 23 19 15 12 11 24 13 0 23 14 10 7 3 0 11 31 31 16 12 9 7 5 3 12 45 25 16 13 11 9 7 7 16 35 23 19 16 13 12 10 11 21 47 33 28 24 22 20 19 21 35
2, 0.1		

2, 0.3		

As you can see above in the larger maze plots, there are squares that are completely black where the states are just surrounded by walls and no goals nearby. This is because of the discount as other states have more value because they are closer to the goal, whereas the discounted reward for these states inside the square are much lower as they are farther from the goal.

Below is a table demonstrating the impact of the PJOG on the result of both Value and Policy iteration with a wall penalty of 50.

Maze	PJOG	Value Iteration	Policy Iteration
1	0.1		

1	0.3		
1	0.9		

It is clear from the paths formed by both Value Iteration and Policy Iteration, that a PJOG of 0.1 or 0.3 does not make a significant impact on the path chosen because even thought a PJOG of 0.3 represents a more stochastic environment it is still not enough to warrant not navigating near a wall if a goal is present. On the other hand, the final path for Value Iteration when the PJOG is 1.0 shows that the agent chose to avoid the wall even though a goal was near since there is a 90% chance that the agent's decision will not be carried out. The same goes for the policy iteration result.

I also ran these experiments with a PJOG of 0.3 and reduced the wall penalty from 50 to 25. The reduced wall penalty resulted in similar performance to when I reduced the PJOG from 0.3 to 0.1. The agent chose paths that were closer to the wall since it would not lose as much if it hit it, even though it was still in a stochastic environment. Additionally, when I ran these experiments with a PJOG of 0.1 and reduced the wall penalty from 50 to 25. The reduced wall penalty resulted in similar performance to when the penalty was 50 with a PJOG of 0.1. The agent chose paths that were closer to the wall since it would not lose as much if it hit it just as it did when the PJOG was 0.1 and it could afford to go near the wall because it knew it would not hit it.

The discount factor is a decaying measure of how far ahead in time the algorithm considers. At a discount factor of 0, the reinforcement learning algorithm will be very greedy and only care about immediate rewards. The usual value to use is around 0.9. There is another breakdown case in some environments, with the discount factor of 1. An infinite horizon can occur, where the agent continues to collect non decaying rewards by oscillating between states and never ending the episode.

Conclusion

In this paper we discussed two algorithms for MDPs, Value Iteration and Policy Iteration. Both require a full knowledge of the world, state space, and model. Policy Iteration is slower because it needs to calculate the entire Markov Chain from a given state. Value Iteration is quick at computing the optimal policy but requires more iterations to converge. Both perform better in more deterministic settings. Increasing the stochasticity, encourages agents to takes safer actions (i.e. staying away from walls).