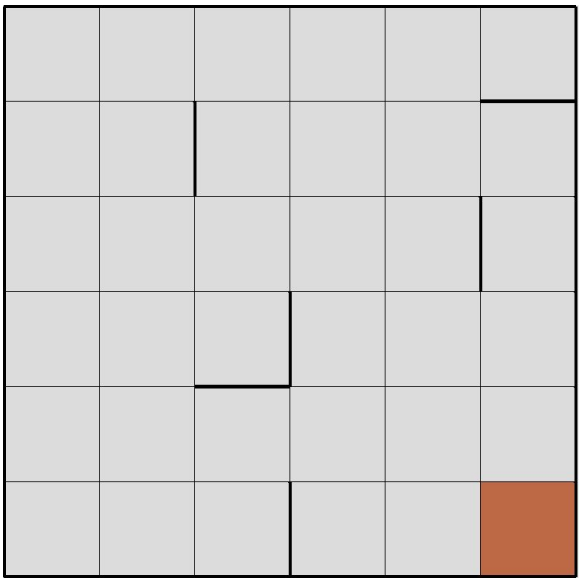
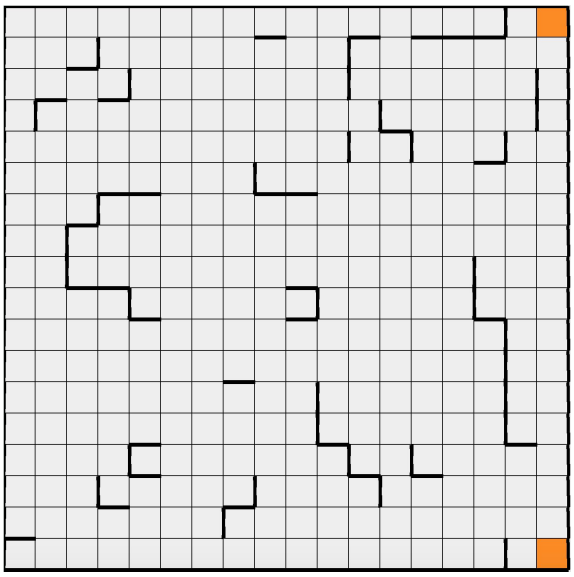
Arthur Shim
ashim6
CS 4641

Analysis

In this assignment, I created two grid MDPS. The first MDP was a 6x6 grid with a single goal state and various walls throughout. The second MDP was a 18x18 grid with two goal states and also various walls. The walls are indicated by solid black lines in the grids. For both of these MDPS, the agent's objective is to reach a goal state as fast as possible. The agent can move up, down, left, and right in each state and gets a reward of -50 each time it runs into a wall.



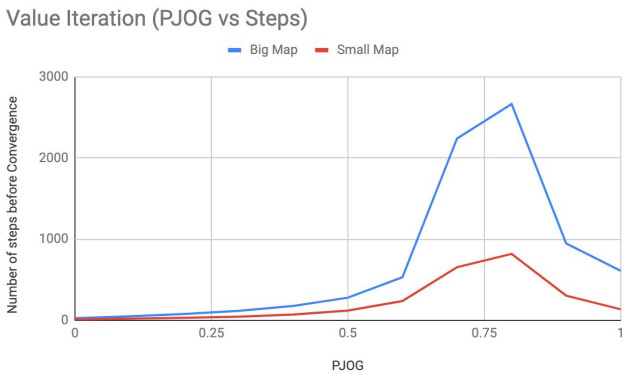


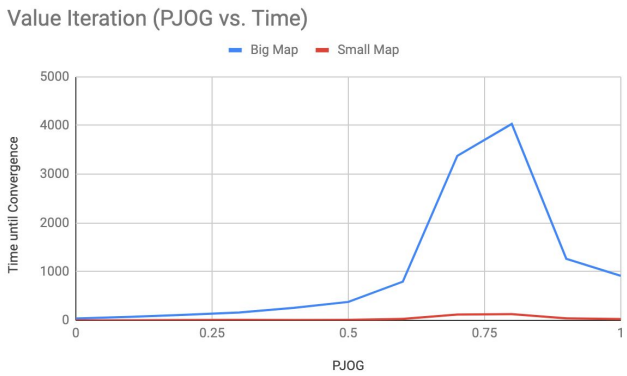
The left MDP therefore has 36 states and right MDP has 324 states. The agent begins in the bottom left cell for both. These MDPs were chosen because it will be interesting to compare value iteration, policy iteration, and reinforcement learning on different sized MDPs. Both MDPs are also designed so that there isn't a straight, clear path from the start state to the goal and there are multiple viable paths from the start to the goal so we'll be able to better test how the algorithms perform in non-trivial problems. The second MDP is also interesting because there are walls very close to the goals, meaning that heading towards any cell near the goal state does not guarantee that the agent is on an optimal path.

Value Iteration

I ran value iteration on both MDPs and tested the number of steps and time they took to converge given different values of PJOG. In these MDPs, PJOG defines the probability that the agent will not take the action it chooses in any state. So if PJOG = 0.3, then there's a 0.3

probability that if the agent chooses to go right, it won't go right and will instead take another random action.

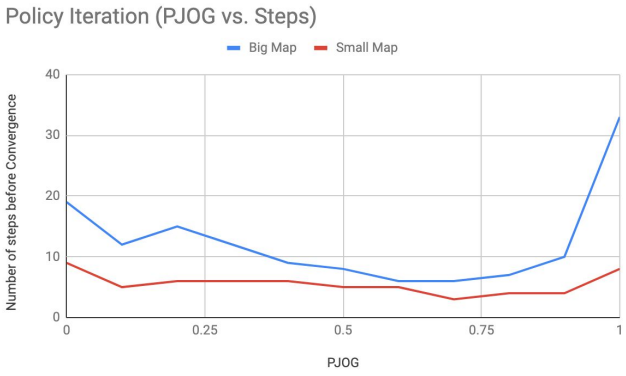


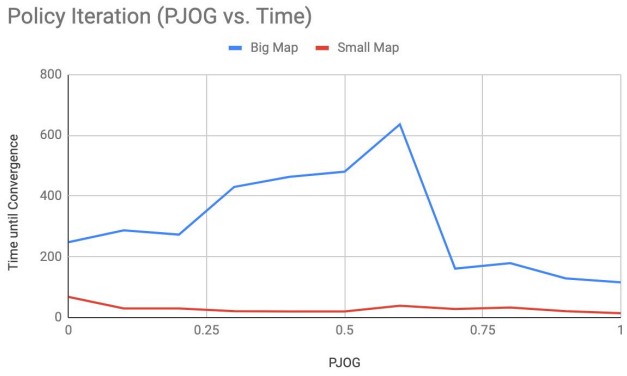


For both time and number of steps, both MDPs showed similar trends. From PJOG = [0, 0.5], both MDPs show steadily increasing times and number of steps. From PJOG = [0.5, 0.8] there was a significant spike in time and number of steps before convergence. This makes sense because for these values of PJOG the agent has a more than 50% chance of not following the action it elects to take, so it cannot reliably be sure of where it will go next and takes much longer for the agent to visit every state and figure out the value of each state. From PJOG = [0.8, 1] the time and number of steps decreases because the agent can reliably not take the action it elects to take at a state. This reliability will cause the agent to require less steps before convergence.

For every value of PJOG, the larger MDP takes more time before converging which makes sense since it has far more states that the smaller MDP and thus the agent will require more actions to visit each state enough times to converge.

Policy Iteration





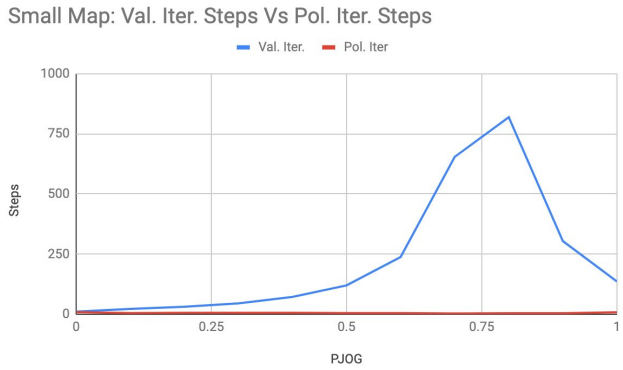
Like value iteration, I tested the run time of policy iteration with respect to different values of PJOG. Overall, policy iteration took many less steps to converge than value iteration, with peaks at PJOG = [0.9, 1]. This is likely due to the fact that when PJOG is high, it takes many

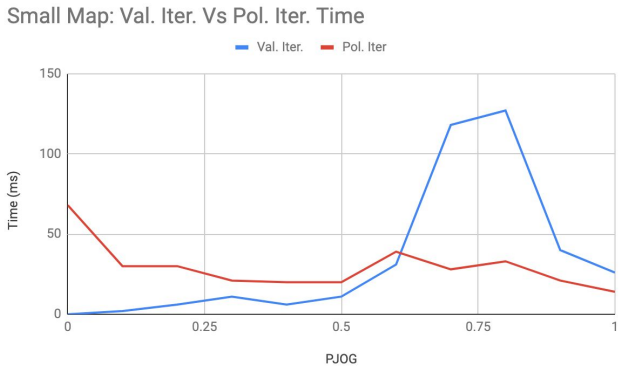
more actions for the agent to fully explore each state enough so that the policy converges, thus explaining why the number of steps increases.

The run time of policy iteration behaved very differently to the number of steps, peaking at around PJOG = 0.6 for the large map. This indicates that although the number of steps before convergence was small at this PJOG value, the time required to complete the steps was higher than for a higher or lower PJOG value. This could be due to the fact that the value function computed at each step was difficult to compute due to the fact that a PJOG value of 0.6 means that the agent cannot reliably be counted upon to take the action that it selects or not take the action it selects. For a lower PJOG value, the agent can be more reliably counted upon to execute the action that its policy dictates, while for higher PJOG values the agent can be more reliably counted upon to not execute the action that its policy dictates. Because the agent cannot be reliably counted upon to do either for a PJOG value of 0.6, the algorithm could have required more time to figure out the value function for each state.

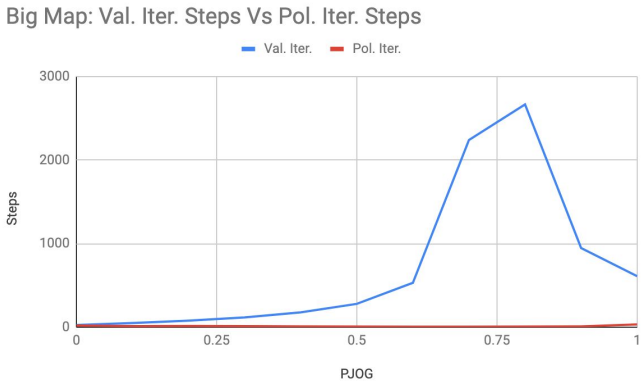
Like value iteration, for larger maps policy iteration took more steps and more time to converge.

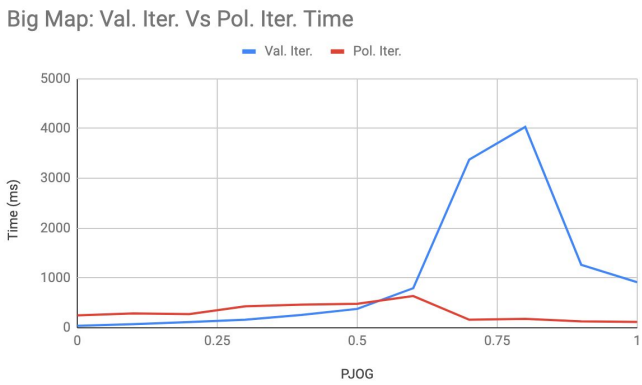
Value Iteration vs. Policy Iteration





Comparing the number of steps before convergence and time before convergence of value iteration and policy iteration on the small map, for number of steps value iteration consistently took more steps than policy iteration. There were only significant differences in the number of steps for each algorithm for higher PJOG values which is likely due to the extra steps the agent needed to visit every state and update their values when the PJOG was high for value iteration. In terms of run time, policy iteration generally took more time than value iteration except for higher PJOG values where value iteration took significantly more time. Policy iteration taking generally more time is likely due to the fact that at each step, value iteration only needs to update the value of each state while policy iteration needs to generate a value function and then update the policy. For a small map with few states, value iteration is faster. However, for higher PJOG values where value iteration took significantly more steps than converge than policy iteration, this led to value iteration also taking more time to converge.

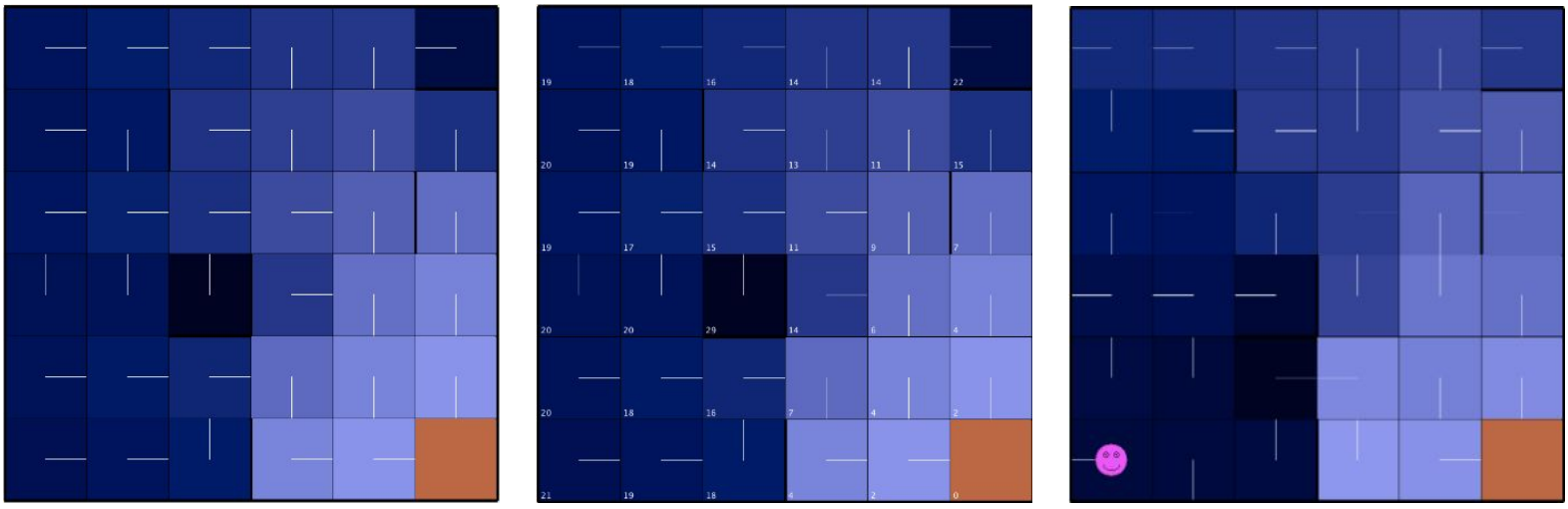




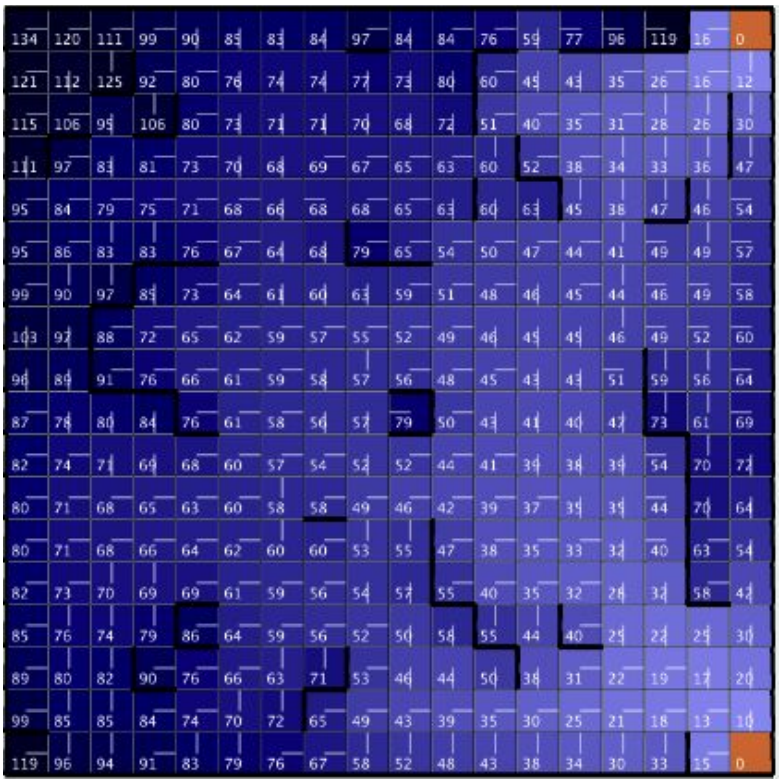
For the big map, the trends in both algorithms remained the same due to the same reasoning as before. However, the overall number of steps and run time for both algorithms was significantly higher, especially for value iteration. This makes sense, since a map with 324 states should take far more time and iterations for the agent to visit each state enough times to compute their values than a map with 36 states.

Q Learning

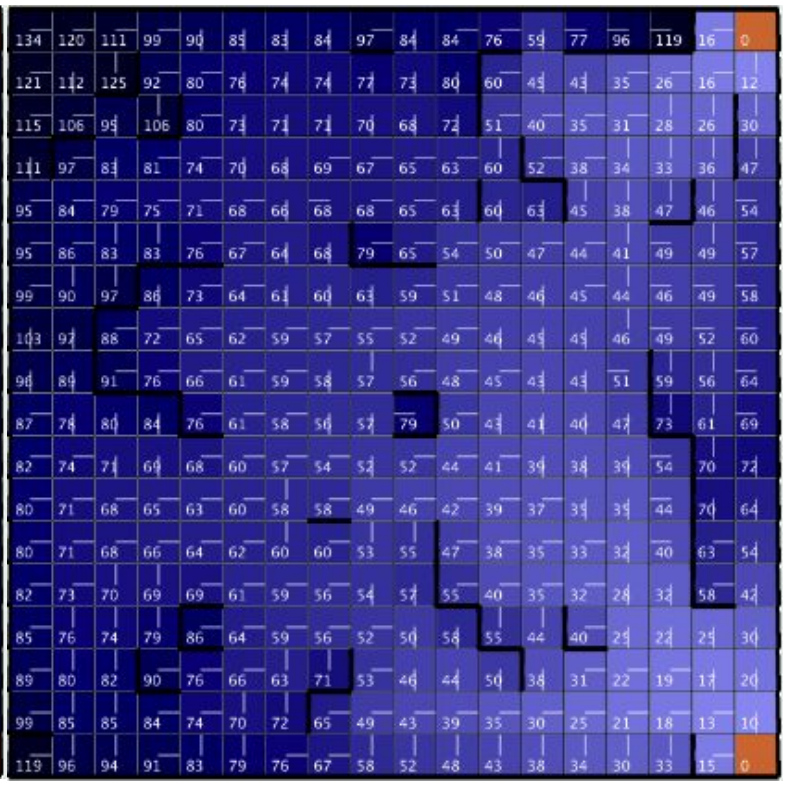
For the final portion of this assignment, I performed Q learning on both MDPs to compare how it performed with value and policy iteration. Below are the solutions generated for the big and small maps by value iteration, policy iteration, and Q learning using PJOG = 0.3:



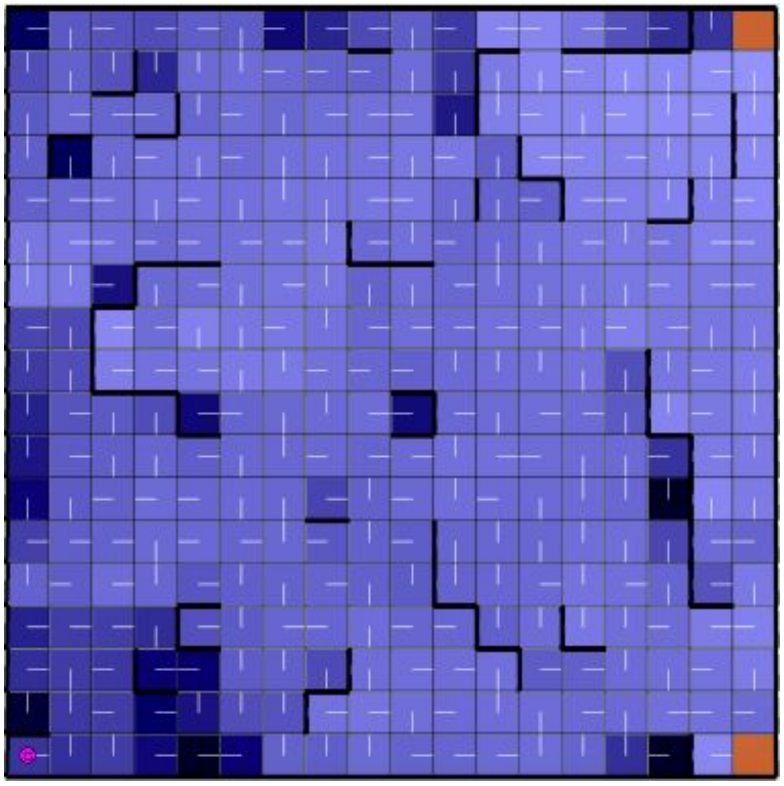
(Left to right: value iteration, policy iteration, Q Learning)



Value Iteration



Policy Iteration



Q Learning

For both MDPs, Q learning took far more iterations than value iteration and policy iteration to converge. After running Q learning for 5000 cycles, the algorithm still could not converge to a solution that was even close to as good as the one produced by value iteration or policy iteration. You can observe in the small map produced by Q learning, the starting state has picked going right as the optimal action, which takes the agent directly into a wall. Most of the states in the small map have going down or going right as the optimal actions, which is the correct direction for the agent to go to move closer to the goal state. However, there are a few states that have the agent go farther away from the goal state. In the large map, there are a significant number of states that tell the agent to go left when both goal states are to the right. The states around the goal states also don't always tell the agent to go into the goal, but in some cases tell the agent to head towards a wall or away from the goal. So even after so many iterations, Q learning was not able to produce an optimal action for every state.

Q learning is a model free reinforcement learning algorithm, and thus is more effective than value iteration or policy iteration when the model isn't known. In this case, we have access to the entire map so value iteration and policy iteration are more effective algorithms in solving these MDPs. Since Q learning uses both exploration and exploitation to find the optimal policy, as the number of states increases the amount of time it takes for Q learning to converge drastically increases in comparison to value iteration and policy iteration. In the first 1000 cycles, Q learning is able to find a pretty good policy for some states, but it needs to perform a lot of exploration before it reaches every state. Because of the randomness involved between exploring and exploiting, it takes Q learning many more iterations to visit each state enough times to determine a good policy for every state than value iteration or policy iteration. If we were to run Q learning for tens of thousands of more cycles, or hundreds of thousands, it would likely be able to converge to a solution as good as the ones produced by value and policy iteration.