# Assignment #4: Markov Decision Processes

Connor Bohan – connor.bohan@gmail.com

## MPD #1: Doors and Keys

### Description

The first MDP is based on the grid world example. It is a 15x18 tile world with four rooms. The first two rooms are connected while the third and fourth rooms are blocked off by doors. The second room has two keys in it which the agent can pick up and use to open the doors. Both keys can unlock both doors, so the agent can get the keys in either order. The agent is trying to get to a chalice in the corner of the fourth room. The agent has four possible actions in each state: north, east, south, and west. Moving onto a space with a key picks up that key. Moving onto a space with a door while the agent has a key opens the door and consumes the key. According to Burlap’s reachability analysis, there are 1035 reachable states.

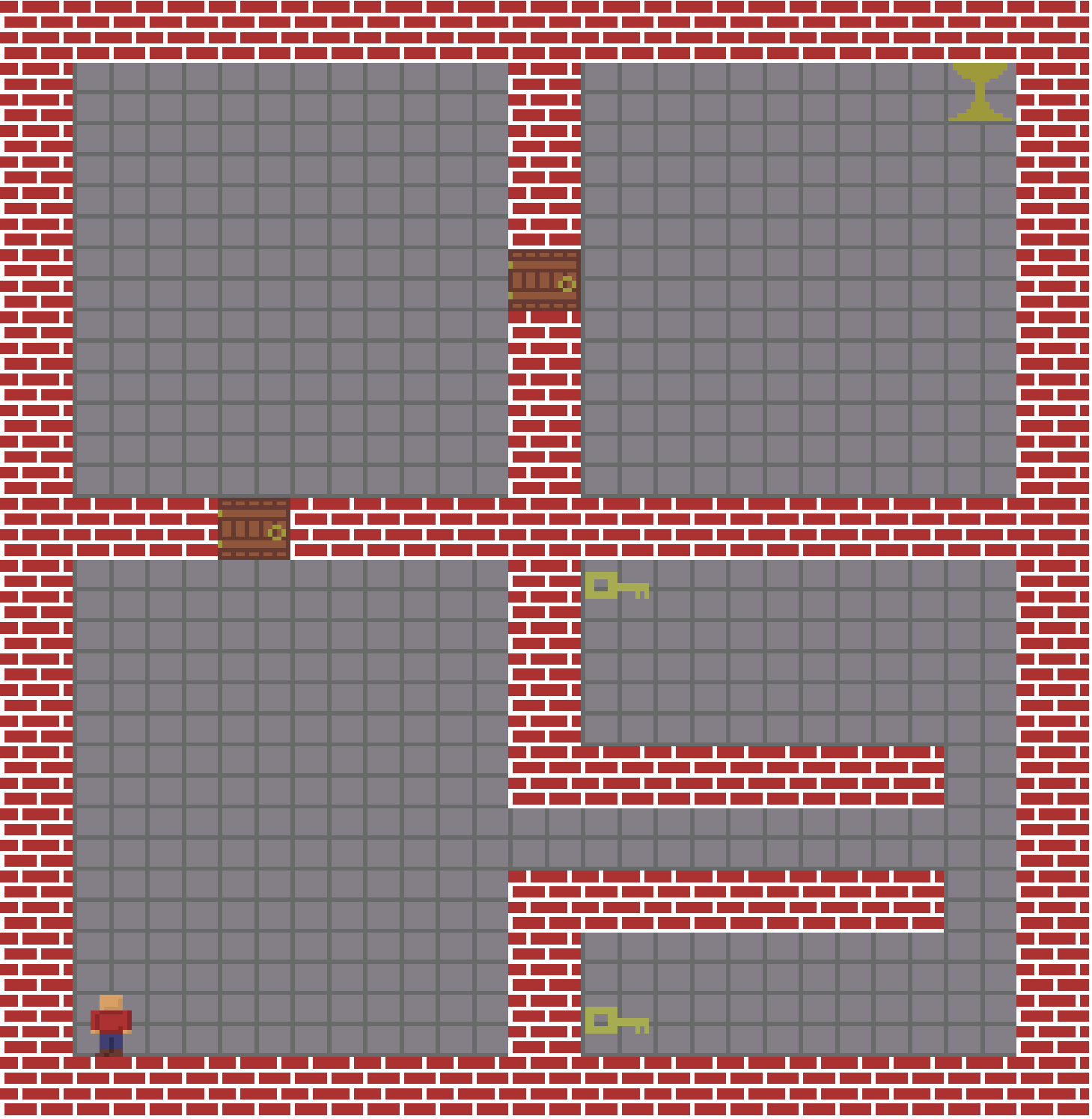


Figure 1: The visualized representation of the Doors and Keys MDP problem. The agent is in the bottom left and the chalice is in the top right.

Part of what makes the Doors and Keys MDP interesting is that it adds a layer of complexity on top of grid world. Both keys can be in one of three state: on the ground, on the agent, or consumed. The doors can be either open or closed. So, while the number of open grid spaces is fairly small, there are actually 36 times that many states (3 states for the first key \* 3 states for the second key \* 2 states for the first door \* 2 states for the second door). The second interesting thing about this MDP is that it has many unreachable states. For example: its impossible to be anywhere past the first door while both keys are still on the ground. I’m not sure what impact, if any, this will have on policy iteration or value iteration. Another interesting characteristic is that there are several ways to get to the reward. The optimal way is to collect both keys first and then unlock both the doors. However, the keys can be collected in either order.

### Value Iteration

I decided to test the Doors and Keys MDP over multiple levels of stochasticity (i.e. the probability that performing an action will result in the intended outcome). When there is no stochasticity, value iteration requires 31 iterations to converge. Convergence took .42 seconds on my machine. When I decreased the probability that the agent would move in the correct direction to 50% (i.e. there is a 50% chance the agent will move in the correct direction and a 16.6% chance the agent will move in any of the other three directions) it took 136 iterations or .98 seconds to converge.

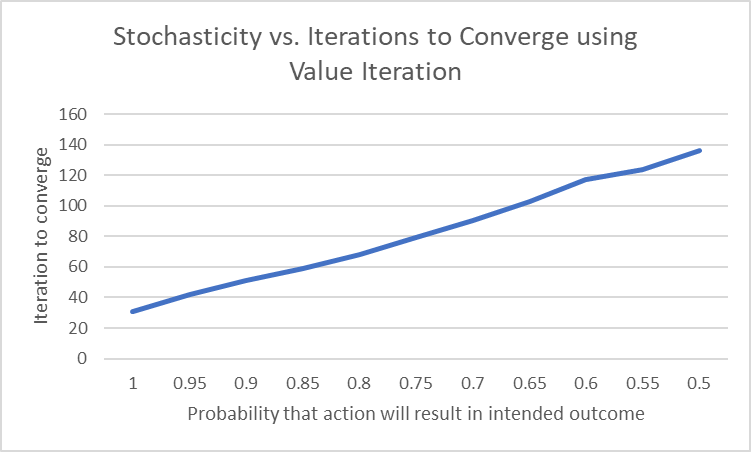


Figure 2: The number of iterations that value iteration needed to converge compared to the probability that the agent moves in its intended direction. As the probability decreases, the number of iterations required seems to increase linearly.

Due to the high dimensionality of this problem it is a little difficult to show the policy that value iteration decided upon. I’ve decided to show four different ‘slices’ of the policy. One slice shows what the agent will do when it has zero keys, there is two slices for when the agent has just one of the keys, and finally there is one slice for when the agent has both keys. An important thing to note is that the way the policies are rendered is flipped vertically from how the visualizer renders the room. In the visualizer the starting room is on the bottom left, while on the policies visualizations is rendered on the top left.

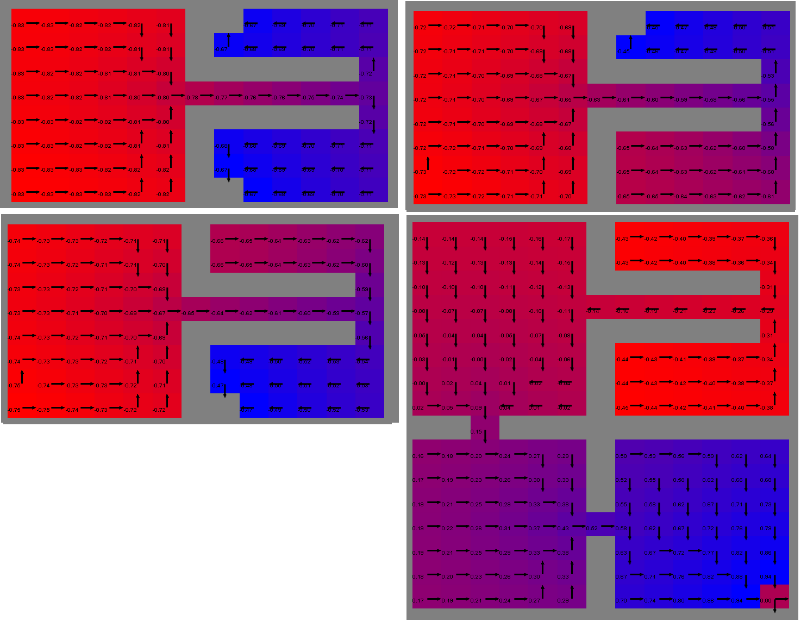


Figure 3: The top left image shows the policy when the agent has zero keys. The top right and bottom left images show the policy when the agent has one key, and the bottom right image shows the policy when the agent has both keys.

From these policy visualizations, we can see that the policy that value iteration settled on was to go get both of the keys and then go to the reward.

### Policy Iteration

Policy Iteration worked… weirdly on this MDP. I made sure to keep the discount factor and max delta the same between the value iteration and policy iteration tests. With zero stochasticity, policy iteration took 33 iterations to converge compared to 31 of value iteration. Based on my understanding, policy iteration should always converge in fewer iterations than value iteration. I’m going to chalk this one up to the black box nature of Burlap’s value and policy iteration methods (possible because VI is updating values in place in a random order?). Another odd characteristic of policy iteration is that it converges in fewer iterations the more stochastic the world is.

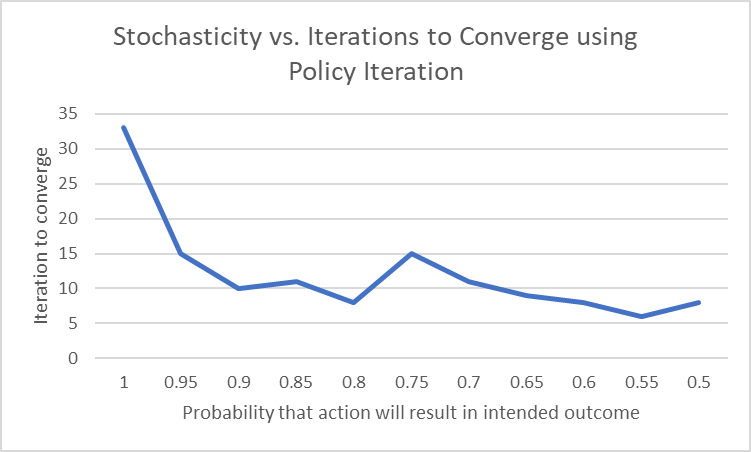


Figure 4: The number of iterations that policy iteration needed to converge compared to the probability that the agent moves in its intended direction.

One of the reasons that policy iteration took fewer iterations to converge when agent movement was more stochastic was that the inner value iteration loop took longer. The inner loop on took around 15 iterations on average when there was no randomness to the agent’s movement. When there was high randomness in the agent’s movement it took around 100 iterations to evaluate the policy. Policy iteration was much slower in clock time that value iteration. It took as little as 4 seconds and as many as 7 seconds to converge on my computer.

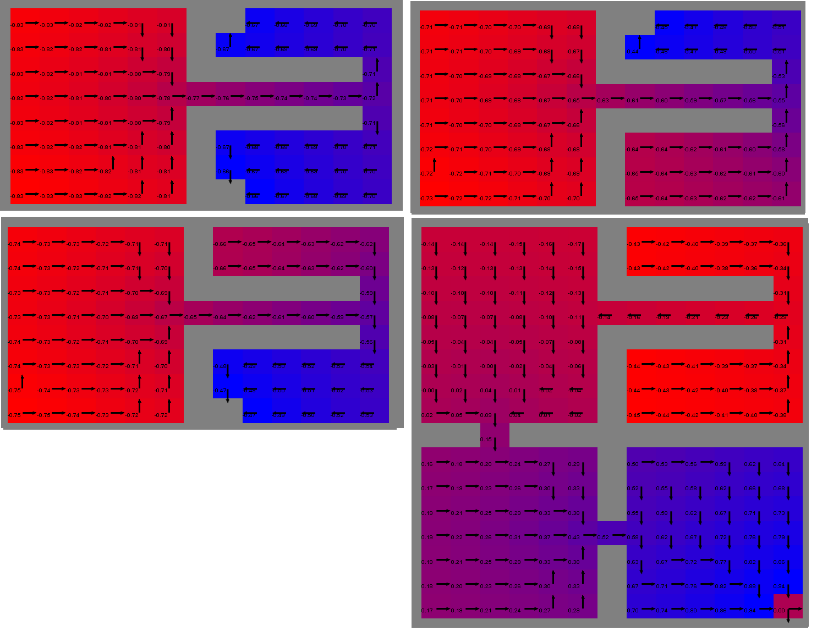


Figure 5: This figure shows the policies that policy iteration chose for when the agent has zero keys, one key, and two keys.

Policy iteration chose a similar policy to value iteration. It decided to get the other key first and there are a few areas where it chose a slightly different, equally valid path through some of the rooms. Overall, I don’t see any advantage to using policy iteration for the Doors and Keys MDP.