Abhishek Deo

Assignment 4

CS 4641

Markov Decision Processes

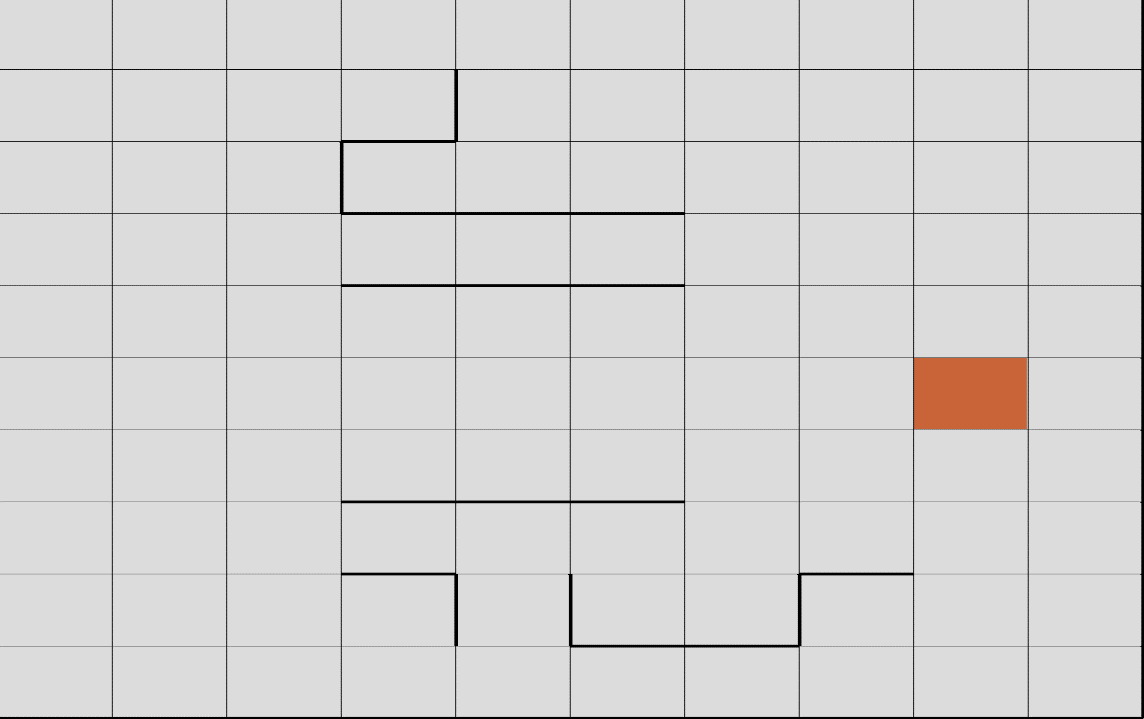
**Intro:**

* Value iteration, policy iteration, and Prioritized Sweeping were used to solve a medium and a relatively large MDP problem.

**Testing Parameters:**

* To determine the optimal policy, value iteration and policy iteration were both used on the medium maze with various modifications on the parameters of the problem. The parameter changed that caused drastic effects in performance was noise in the environment. A larger value would result in more noise and vice versa. I incremented the PJOG parameter by 0.1 up till a 1 for maximum noise. (PJOG is the noise parameter)

**Problem One:**



* The first maze used to test is a 10x10 medium class maze with a few obstructions (traps) with one goal. The location of the goal is interesting as there is a direct way to get to the goal without ever acknowledging the location of the obstructions. However, the obstructions placed are a bit complicated to traverse and can pose significant problems for and agent. There are a total of 100 states and running into a wall incurs a cost of 50 points.

**Value Iteration**

|  |  |  |
| --- | --- | --- |
| Noise (PJOG) | Time (seconds) | Steps |
| 0.1 | 0.1 | 28 |
| 0.2 | 0.06 | 41 |
| 0.3 | 0.18 | 61 |
| 0.4 | 0.23 | 99 |
| 0.5 | 0.34 | 164 |
| 0.6 | 0.25 | 164 |
| 0.7 | 0.52 | 304 |
| 0.8 | 1.91 | 1096 |
| 0.9 | 0.92 | 480 |
| 1.0 | 0.98 | 321 |

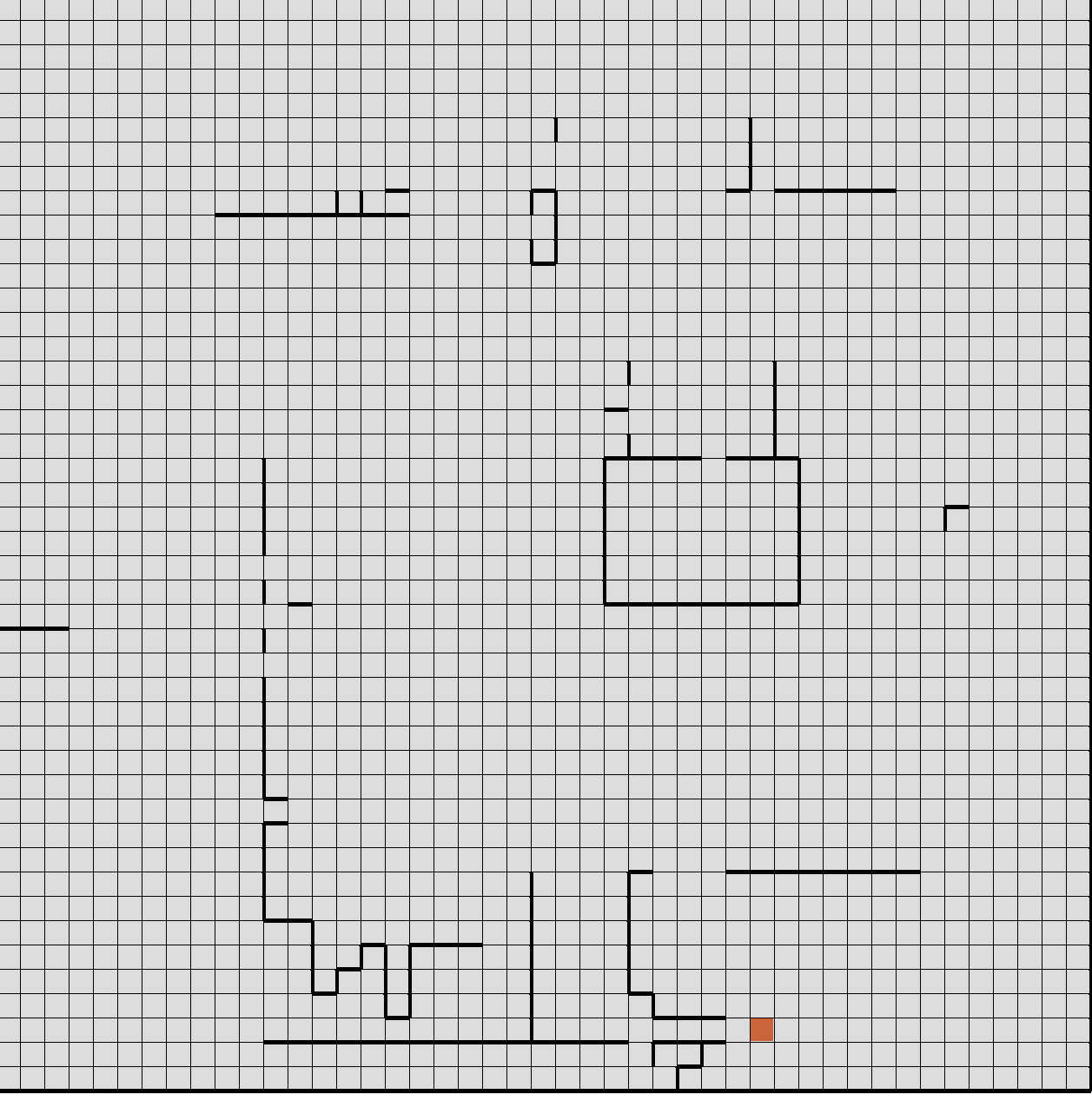
* The results of the number of steps vs time taken for value iteration on the medium maze are really interesting. The table above displays the number of steps taken to converge varied with the amount of noise incremented by 0.1. As the amount of noise increases, the amount of time taken increases as well but from the table there are a few anomalies that do not represent this completely. Specifically at the point where PJOG=0.1 and PJOG=1.2 and then after PJOG=0.8. Having one goal state most likely caused this issue as having multiple goal states minimizes time and the number of steps. Because there was one goal, due to the nature of the algorithm it is by chance that there were better results at PJOG=0.2 compared to PJOG=0.2. Also the significant difference between PJOG=0.8 and PJOG=0.9 reflects the same instance that happened from the first two runs but at a larger scale.

**Policy Iteration**

|  |  |  |
| --- | --- | --- |
| Noise (PJOG) | Time (seconds) | Steps |
| 0.1 | 0.5 | 6 |
| 0.2 | 0.58 | 7 |
| 0.3 | 0.27 | 7 |
| 0.4 | 0.31 | 6 |
| 0.5 | 0.46 | 5 |
| 0.6 | 0.68 | 6 |
| 0.7 | 0.95 | 4 |
| 0.8 | 0.36 | 3 |
| 0.9 | 0.18 | 7 |
| 1.0 | 0.14 | 19 |

* The results of the policy iteration test runs using the same variation in the noise variable is different from the results of value iteration. The nature of policy iteration evaluates the action of your process at each iteration to improve the control policy which allows for faster run times at least for this maze. It was significantly quicker than value iteration and took a significantly fewer amount of steps at each noise increment. One thing to note is that the pattern at least in terms of time was very similar to the pattern followed by value iteration but on a smaller scale. One can notice the big jump backwards in time and steps from 0.7 and 0.8 where it means that there was less uncertainty on what decision to make. This makes sense considering that there is a straight path to the end goal without ever encountering the obstacles. The other notable difference is the number of steps required to reach the goal in combination with the time required compared to the other tests. It seems that at 100% noise the number of steps increases but the time required is significantly shorter. This is most likely because the value in any adjacent cell is directly calculated and disregarded if the value cannot be improved by changing the action taken under the current policy. With high noise input this step becomes stricter and improves the performance of the policy.

**Problem 2:**

****

* The first maze used to test is a 45x45 large class maze with numerous obstructions (traps) and one goal. Compared to the small maze, the location of the goal is very interesting as it is almost surrounded by obstructions and is close to the edge wall. Even more interesting is the complexity of traps and their setup on the grid. The traps that are very complicated are concentrated together and close to the goal. The one trap near the center with a one cell entrance and exit also seems like it is the most dangerous trap on the grid. While the traps on the top of the grid are not as complicated, it is strong enough to throw off any agent for a decent amount of time. There is also a 50 point cost for hitting a wall.

**Value Iteration**

|  |  |  |
| --- | --- | --- |
| Noise (PJOG) | Time (seconds) | Steps |
| 0.1 | 18.32 | 106 |
| 0.2 | 24.27 | 136 |
| 0.3 | 30.51 | 181 |
| 0.4 | 44.51 | 260 |
| 0.5 | 78.16 | 428 |
| 0.6 | 169.11 | 956 |
| 0.7 | 890.44 | 4974 |
| 0.8 | 1080.85 | 5769 |
| 0.9 | 221.31 | 1226 |
| 1.0 | 170.31 | 960 |

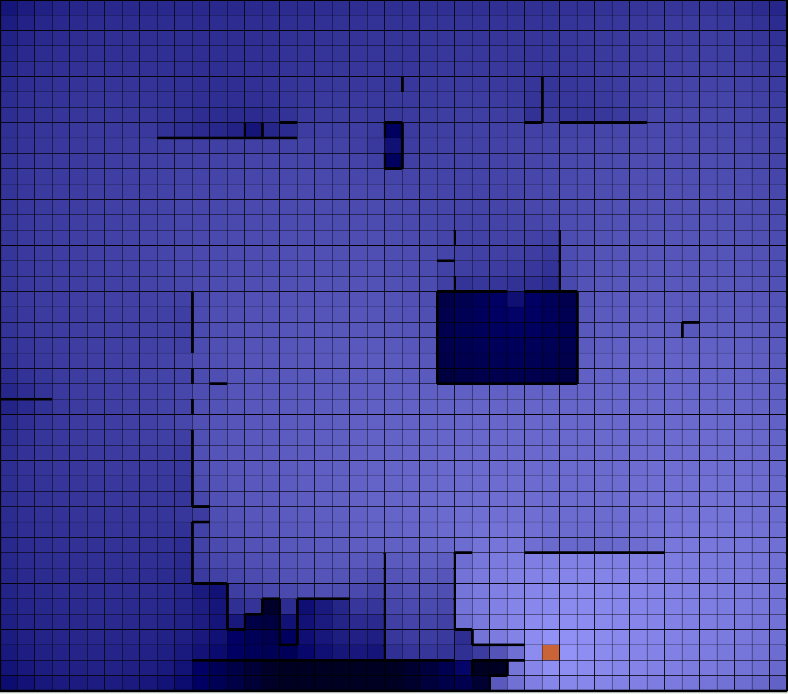
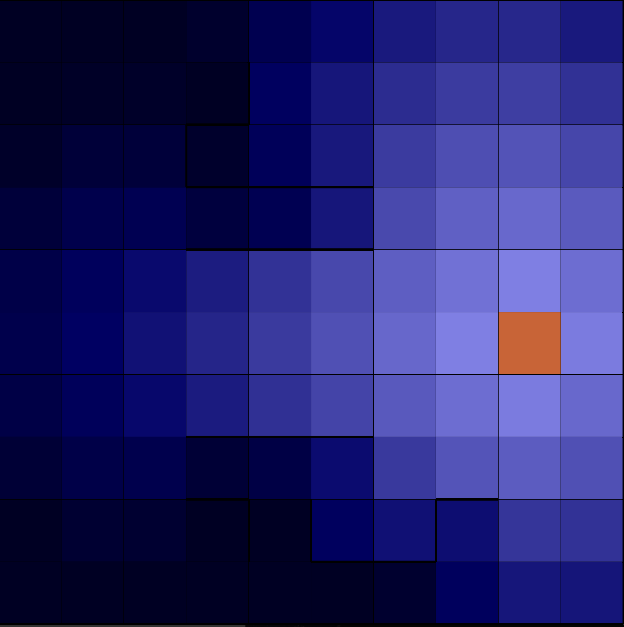
* The results for value iteration were very interesting in terms of both time and steps. The biggest surprise for me was the large difference between test 0.6 and 0.7. There was almost a 530% increase in time and steps required to find the goal. The results followed a similar pattern to the medium class maze results. Similar to how there was only one goal for medium maze, having one goal for the large maze especially considering it was surrounded by walls really affected the time. If there were multiple goals the time required may be reduced by almost a factor of two based on the alignment of the obstructions.

**Policy Iteration**

|  |  |  |
| --- | --- | --- |
| Noise (PJOG) | Time (seconds) | Steps |
| 0.1 | 96.34 | 49 |
| 0.2 | 118.13 | 17 |
| 0.3 | 118.04 | 17 |
| 0.4 | 100.64 | 13 |
| 0.5 | 125.39 | 15 |
| 0.6 | 153.07 | 8 |
| 0.7 | 198.53 | 9 |
| 0.8 | 211.55 | 9 |
| 0.9 | 19.63 | 161 |
| 1.0 | 16.20 | 139 |

* The results for policy iteration on the large maze are very surprising and variance was larger for the steps column compared to policy iteration on the medium class maze. Based on the pattern from the previous test results, I expected very similar step size for almost all iterations of Noise except for maybe the last two (0.9 and 1.0). There is a big difference in the number of steps required to take at 10% noise versus 20% noise. The best explanation for this observation is the placement of the goal in relation with the traps. The results for this maze is on a much larger scale compared to the medium class maze due to primarily its complexity rather than its size. The large jump from 9 steps to 161 steps and the decrease in time shows an inverse relationship on how strong noise can affect space complexity and time complexity. This is also due to the same reason that explains the inverse relationship in the jump of policy iteration on the medium class maze. Also given the trap near the center and how it’s designed it most likely played a large role and had a detrimental impact on the various time and step results.

**Comparison (The graphs are below the images.)**

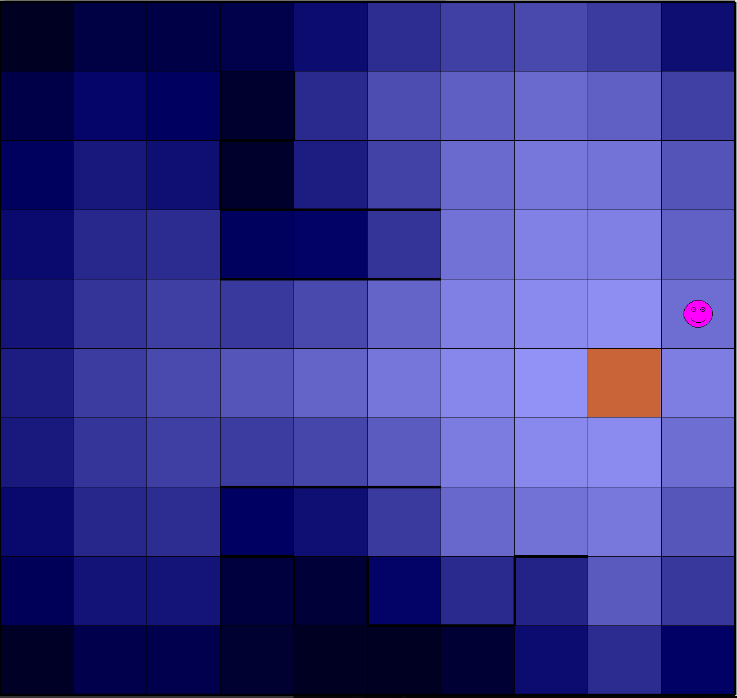
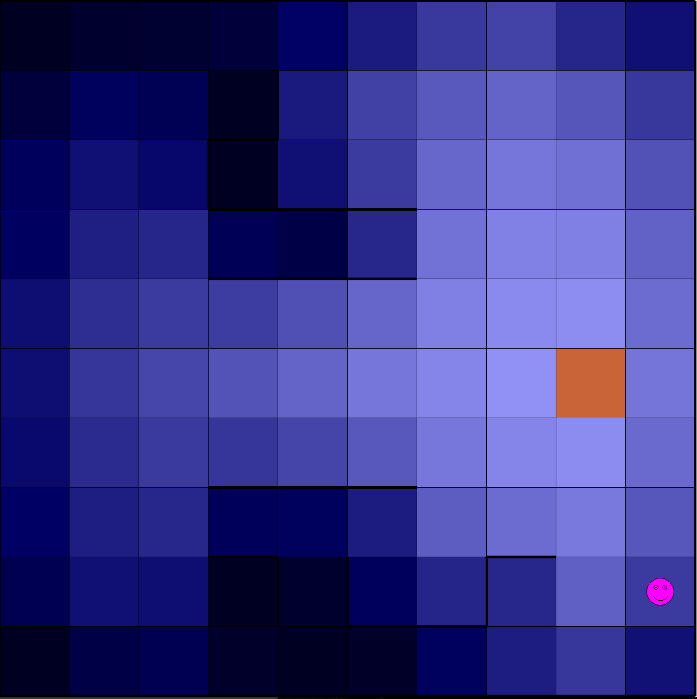


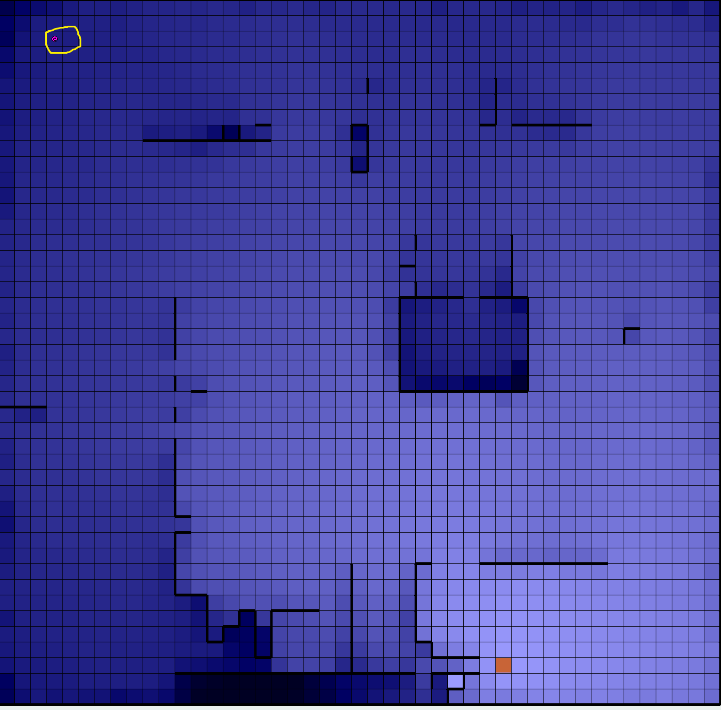
The pictures above are the end results of value iteration at PJOG=0.7. For both of them you can tell that around the more complicated traps the pixels are darker which means there is a higher cost value of going to that state. For the larger maze, most of the grid is rather light but the trap near the center is almost pitch black due to its design. The traps next to the goal at the bottom are also very dark and reflect how complicated it was to traverse them.

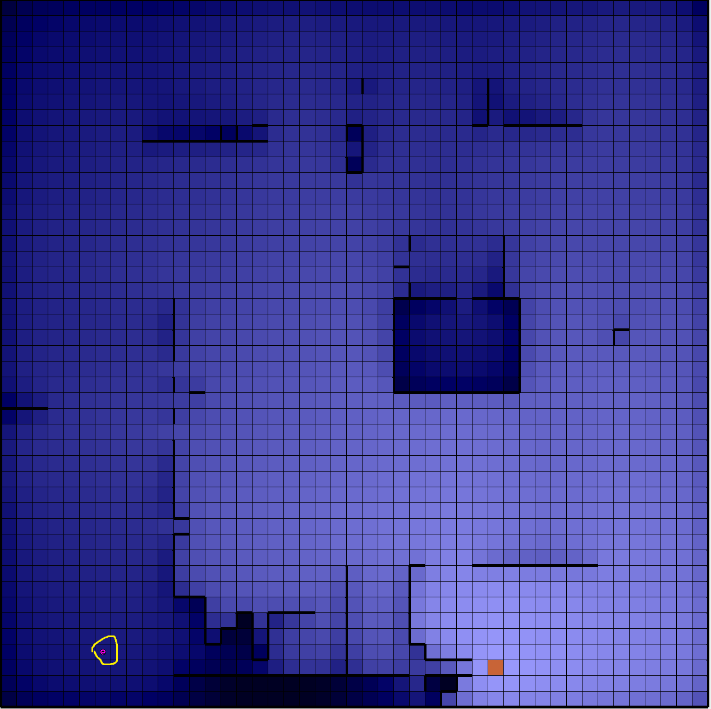
The graphs illustrate well the flow and consistency between both value and policy iteration in terms of time and steps. Major differences for time between the large and medium maze occurred with policy iteration. While policy iteration was more linear in the larger maze, starting from around 30% noise (PJOG=0.3), the time required increased slowly but consistently because the optimal policy was harder to detect. The parameters for deciding the policy is very stringent and thus is easier with more noise. After 0.7, the time required to reach the goal for policy iteration decreases significantly in all cases. Value iteration has a very similar pattern for both time and steps. After around 0.6 and 0.7 there is a huge difference in performance which significantly increases the amount of time and steps required to reach the goal (ignoring the fact that the larger maze points are larger by a significant magnitude). Overall, policy iteration seems to run better for both mazes in both the time and steps metric. The only cases value iteration performed better was for time in and up till roughly 50-70% noise. Because there is only one goal and value iteration tends to work recursively and doesn’t know when to stop unless a stopping criteria is integrated such as a greedy policy as a function, it fared worse than policy iteration. The percent increase between the medium and large maze for time at 80% noise for value iteration is 566%. For steps at 80% noise the percent increase is at 5.27%. I chose PJOG=0.8 for a base comparison because all four graphs showed a significant gap from the previous data point.

**Prioritized Sweeping**

* Prioritized sweeping uses all previous experiences to both prioritize important dynamic programming sweeps and guide the exploration of state-space. Prioritized sweeping is strong in larger state-spaces because it takes advantage of memory-based learning. It makes a decision based on whether to explore the subsequent state or exploit the current state. The variable in change for testing was the epsilon value which influences the probability of exploration. The higher the value of epsilon, the higher the probability of exploring and vice versa.



* The picture on the left shows the result after a 1000 cycles where epsilon is at 0.5 and the picture on the left is where epsilon is at 1.0. Based on the end location, choosing whether to explore or exploit did not make much of difference for the medium class maze because it only consists of 100 states. At epsilon=0.5, where there was roughly a 50% chance it would explore and 50% chance it would exploit, it ended up relatively close to the goal state. At epsilon=1.0, the agent was diagonal to the goal state because it explored on every possible step. There is an offset of 4 squares but very similar to each other. When there was a higher chance of exploiting each state rather than exploring the results were slightly worse but negligible in the sense that it was off by less than 2spaces. Therefore, when the state space is small, especially considering that p-sweeping uses memory-based learning, the decision between exploring and exploiting does not make much of a difference. Now let’s see the effect on a much larger state space with more complicated traps.



* In a much larger state space (45x45) choosing whether to exploit or explore makes a significant impact on run time. Like the medium-class maze example, both the image on the left is at epsilon=0.5 and the image on the right is at epsilon=1.0 after 1000 cycles. The interesting thing about this maze in particular is that it contradicts the point where the higher the chance the agent chooses to explore the closer the agent will be to the goal (or the better the performance). In this case, on the image on the right where the agent only explored resulted in a surprising anomaly. However, it is important to point out the location of the traps in relation to the goal. When the agent explored and exploited with equal chance, it ended up being stuck surrounded by the traps in the bottom left corner. Really, the only thing blocking it and the goal were a few walls but that makes a huge difference. For the case of exploring a 100% of the time, the agent happened to end up far away from many of the traps. Also, considering that the goal state is essentially surrounded by walls with a few access points really makes this maze difficult. Therefore, even if the first case where the agent exploited and ended up closer to the goal, it is in a worse state than the second case. Testing this maze with 100% exploiting resulted in a terrible state where the agent became stuck because it does not learn and the policy became more and more chaotic. While in both cases the agent was still pretty off, the policy was somewhat bad because it seems that the agent learned but did not use what it knew already. After seeing the iterations, it learned of the traps of the corners and went back up to where it started and repeated the same process because it could not determine a more effective approach. Because of this maze, exploration seemed to be a little better but in general, a middle ground of exploring and exploiting would be the optimal choice.