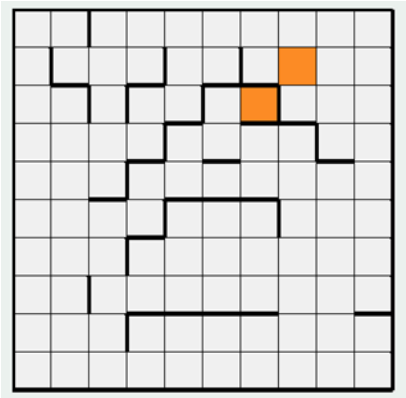
Assignment 4

Name: Shengrui Lyu GTID: 903392423

Introduction to the two MDPs:

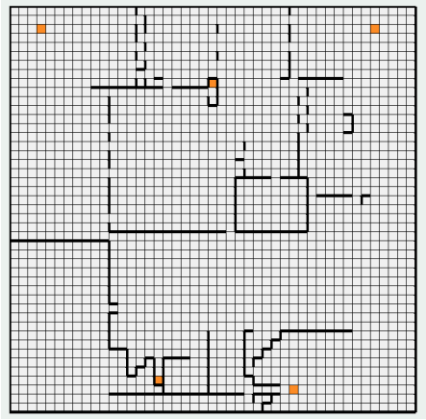
The two MDPs are both grid world problems. The agent will get -1 rewards in each cell and get -50 as penalty when hit the wall. And the cell in orange is the goal, when the agent gets to the goal, the game will terminate. Therefore, the agent should choose the path to get to the get to the goal as soon as possible while avoid hitting to the wall. As for the action model, there is a parameter called PJOG, which indicates the model noise in the environment. For example, PJOG = 0.3 means that 70% of the times an action produces the intended next state, and 30% of the times the action causes the agent to get pushed against its will to any of its neighboring states. We can use this parameter to adjust the action model. And our agent will start at the left bottom corner.

MAZE1:



The MAZE1 is relatively small (13 x 13), and there 169 states in total. It is interesting because of the following reasons: Firstly, there are two goals that are very close to each other, however they are separate by walls. Secondly, although the top right goal is further in direct distance, it has less wall around it. It is interesting to see the agent choice between “risky but close” and “safe but faraway”.

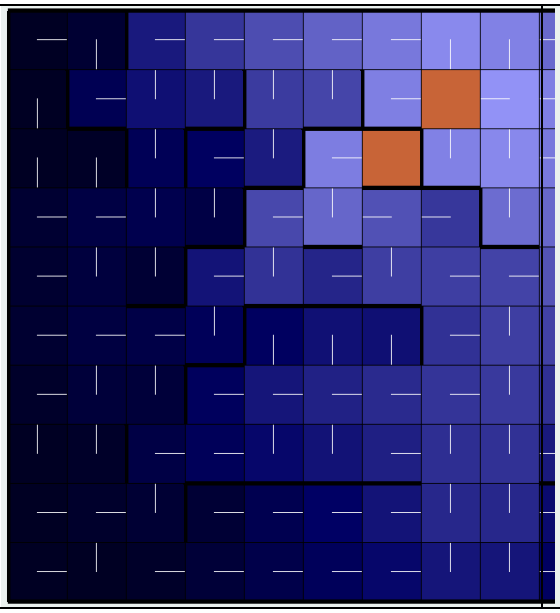
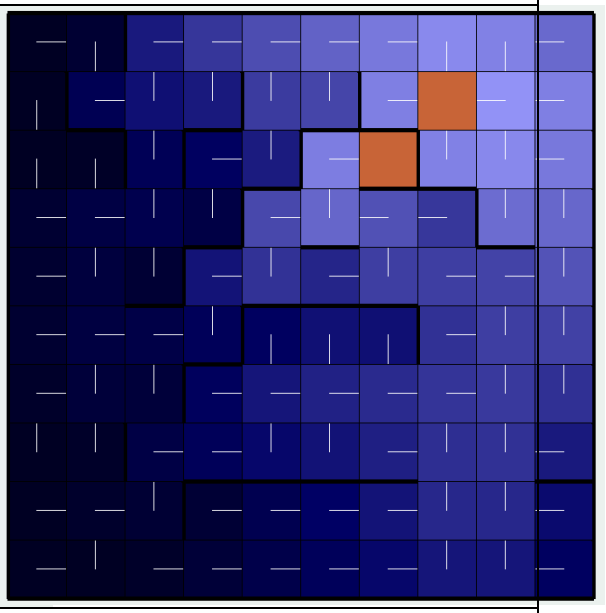
MAZE2:

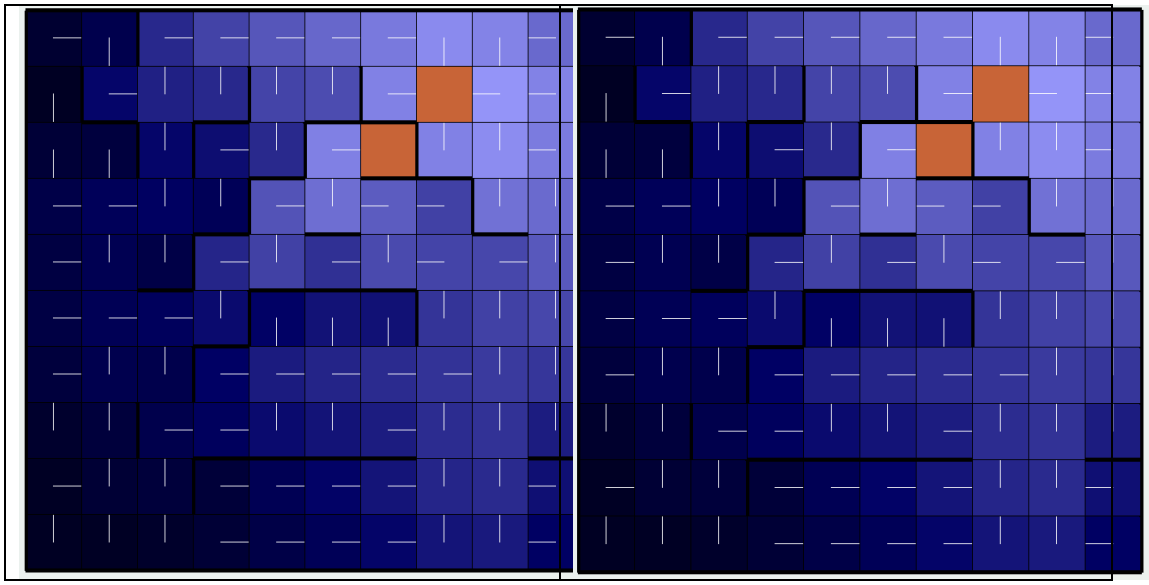


MAZE2 is relatively large(45 x 45), and there are 2025 states in total. There are 5 goal in the map. It is interesting that two goals are very risky (top middle one and bottom left one), especially when PJOG is high. And also it is interesting to compare the performance between the MAZE1 and MAZE2 to see how the size of states space will effect the performance of RL.

Solution to MAZE1:

Maze	Algorithm	PJOG	Total Time	Iterations Needed
1	Value Iteration	0.3	65ms	84
1	Policy Iteration	0.3	122ms	7
1	Value Iteration	0.1	39ms	42
1	Policy Iteration	0.1	162ms	6

Value Iteration	Policy Iteration
Maze 1. PJOG = 0.3	
	
Maze 1. PJOG = 0.1	



(Note: Darker color means higher U(s))

Analysis:

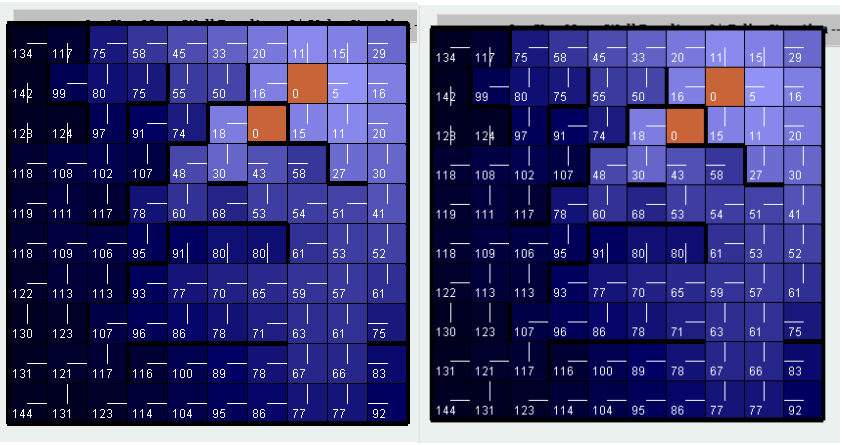
From the above table we can see that:

1. In this MDP, the value iteration takes more iteration than the policy iteration. However, it converges faster. I think this is because the value iteration moves in the value space, and the policy iteration moves in the policy space. Therefore, policy iteration takes larger steps in one iteration by having a policy assumption at the beginning. And policy iteration changes the nonlinear max function in Bellman equation to linear sum by this assumption. But the cost is that policy iteration need to take more time to loop over the states to update the policy after calculating the utilities of the states.
2. They do converge to the same answer. That's because we have the fix reward function and there exist an optimal action for each state. And if there is an action state pair that is not optimal in value iteration or policy iteration, the termination requirement will not be met, and the algorithm will keep updating until all the actions are optimal.

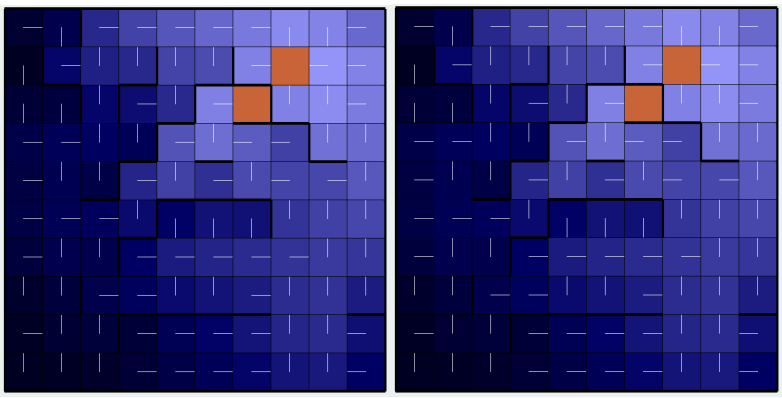
Adjusting parameters:

1. Transition model: As mentioned in introduction, the transition model depends on PJOG which indicates the model noise in the environment. For example, PJOG = 0.3 means that 70% of the times an action produces the intended next state, and 30% of the times the action causes the agent to get pushed against its will to any of its neighboring states. From the above table, after increasing the PJOG, the U(s) for cells along the path to the left bottom goal decreases, and the optimal action tends to move away from this path. It is because this path is very risky, it is surrounded by walls, and a higher PJOG means that it is more likely for the agent to hit the wall along this path.

2. Reward Function + Transition model: I changed the penalty for hitting the wall to -100. And from the following two graphs we can see that the state value for the cell around the wall is decrease a little bit. By the way, the policy iteration and value iteration still give the same answer. (Left is the answer for value iteration, Right is the answer for policy iteration). Also, we can see that the difference between penalty = -100 and penalty = -50 can be seen more clearly under PJOG = 0.3. Because when PJOG =0.3, it is more likely for the agent to hit the wall when it reaches the cell near the wall.

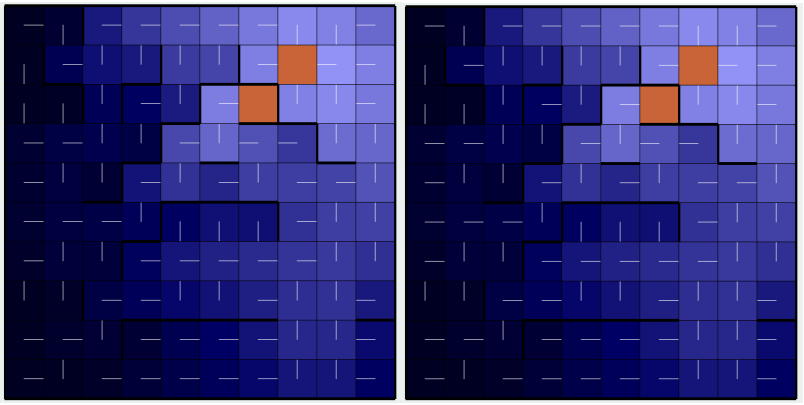


Value Iteration(left) and Policy Iteration(right) PJOG =0.3, Penalty =-100



Value Iteration(left) and Policy Iteration(right) PJOG =0.1, Penalty =-100

3. Discount: Ideally, a lower discount factor will result in an answer that prefer the shorter path to the goal. But we can not see it clearly in this problem, I think it is because of the following two reasons: Firstly, the maze is too small, and secondly the path to the goal is “shaped” by the wall, so the wall is still the dominant constraint. We can see that after change the discount rate from 1 to 0.5, the answer does not change for value iteration and policy iteration. (PJOG = 0.1, left is policy, right is value)

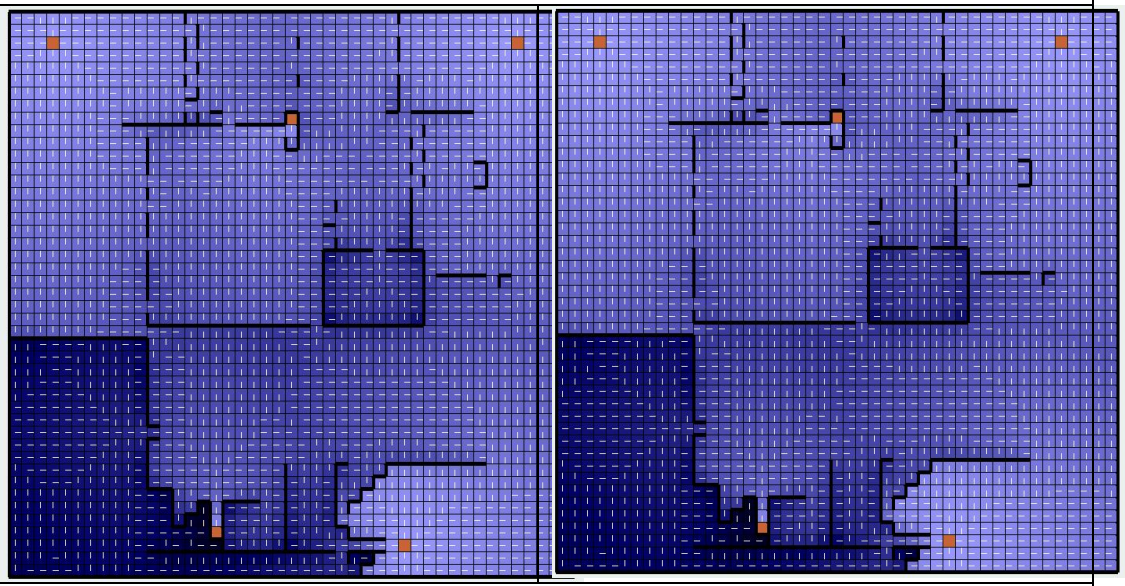


4. State representation: I don't think it is a good choice to change the state representation here in this problem. The cells are fixed for the problem, and actions are also fixed. This state action pair is trivial, but it is effective and universally used in RL research and teaching.

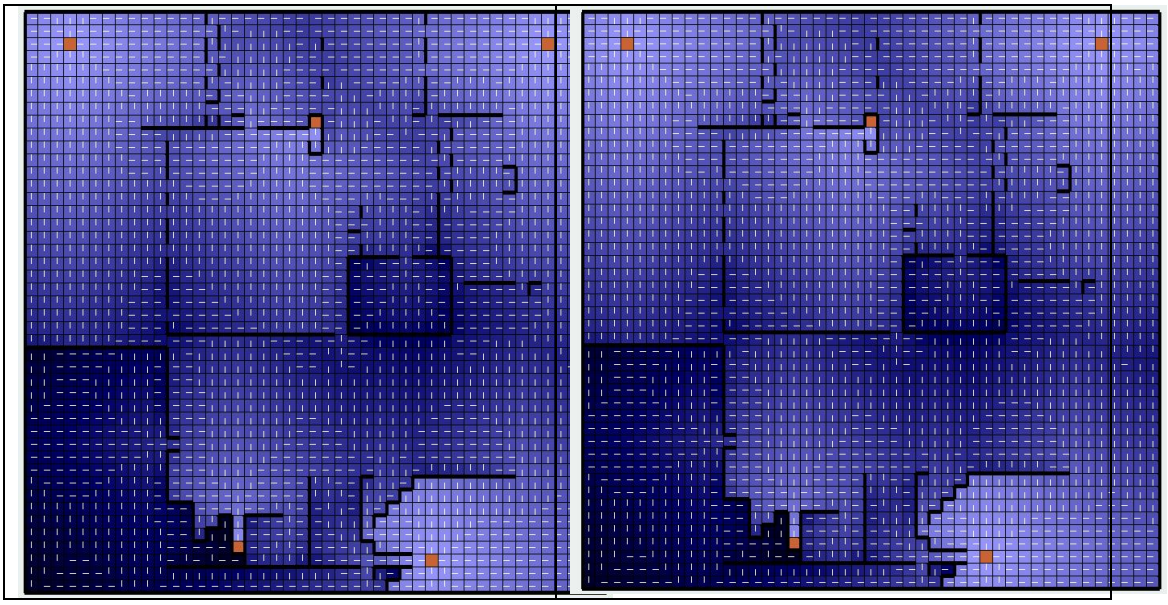
Solution to MAZE2:

Maze	Algorithm	PJOG	Total Time	Iterations Needed
2	Value Iteration	0.3	13063ms	154
2	Policy Iteration	0.3	48438ms	16
2	Value Iteration	0.1	5595ms	66
2	Policy Iteration	0.1	42204ms	22

Maze 2. PJOG = 0.3



Maze 2. PJOG = 0.1



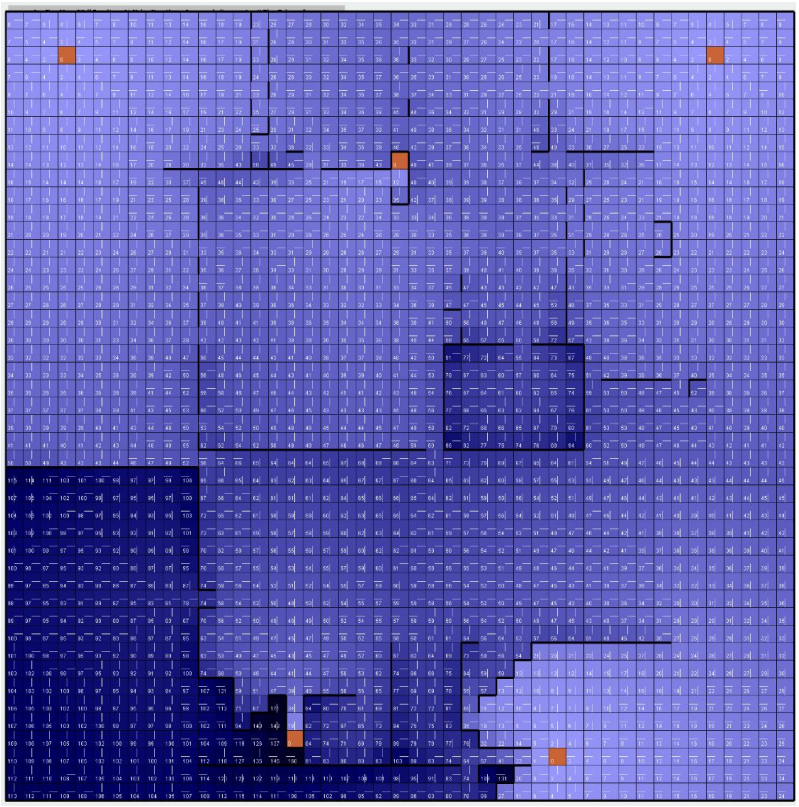
Analysis:

1. As mentioned in MAZE1 result, the value iteration uses less time to solve the problem. And the gap in time used is larger in this problem, because the states in MAZE2 is more than the states in MAZE1. The utility updating loops run over all the states, therefore policy iteration will take a longer time.
2. As mentioned in MAZE1 result, the answer for the value iteration and policy iteration is the same. And the reason is also mentioned there.
3. Compared to MAZE1, MAZE2 has more states. And it will take more time and more iteration to solve. Interestingly, the large maze has about 12 times the number of states that the small maze does, but policy iteration took almost 300 times longer on it, timewise.

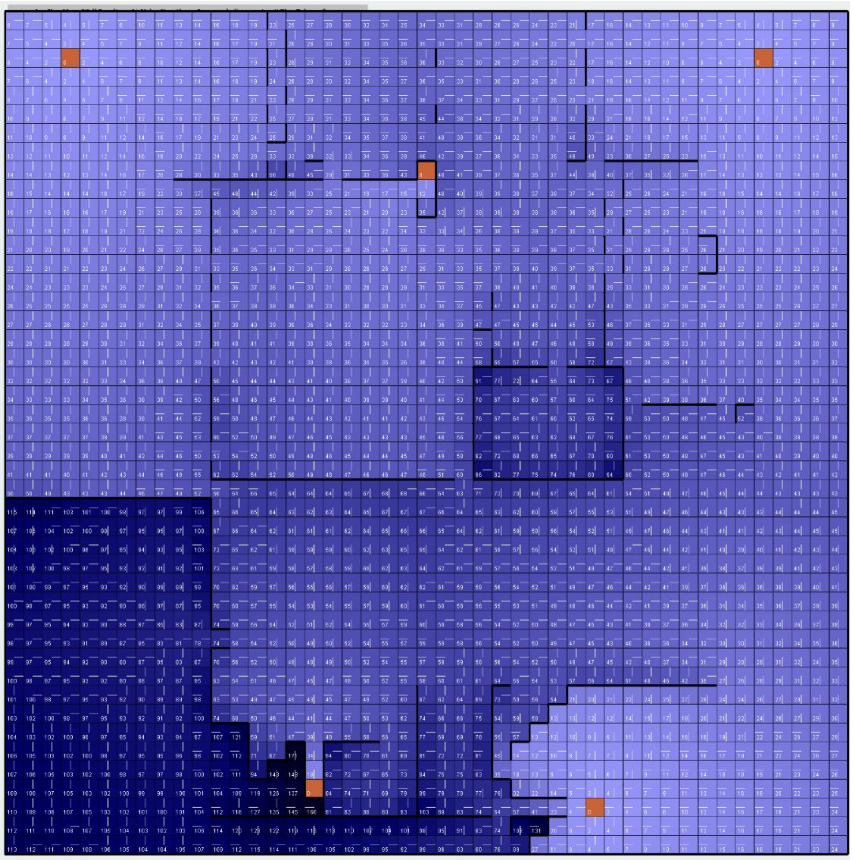
Adjusting parameters:

1. Transition model: As we have more states here in MAZE2, we can see clearly when PJOG is 0.1, the states around the wall have higher value than its value when PJOG is 0.3, because it is less risky to be around the wall when PJOG is lower.
2. Reward function + Transition model: We can compare the following graphs with the graphs above. After we change the penalty for hitting the wall to -100, the color around the wall become lighter, which means the state value for these cells decreased, because the penalty for hitting the wall become higher. Also, we can see that this change can been seen more clearly when PJOG = 0.3, because it more likely for the agent to hit the wall when it reaches the cell near the wall.

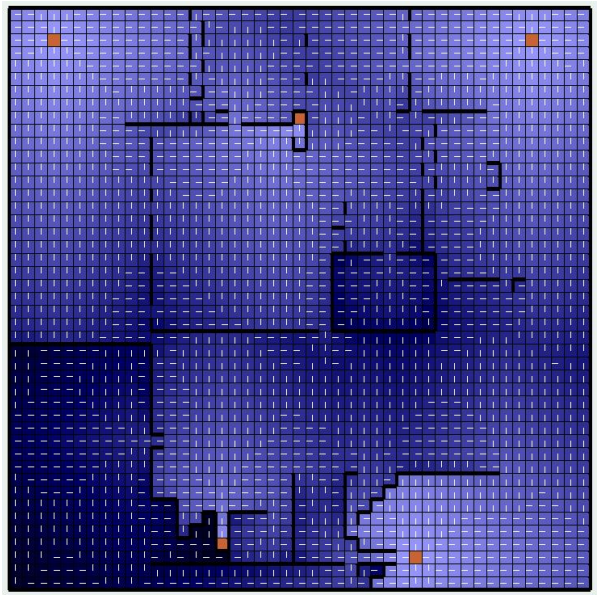
Value iteration result (penalty for hitting the wall = 100, PJOG = 0.3)



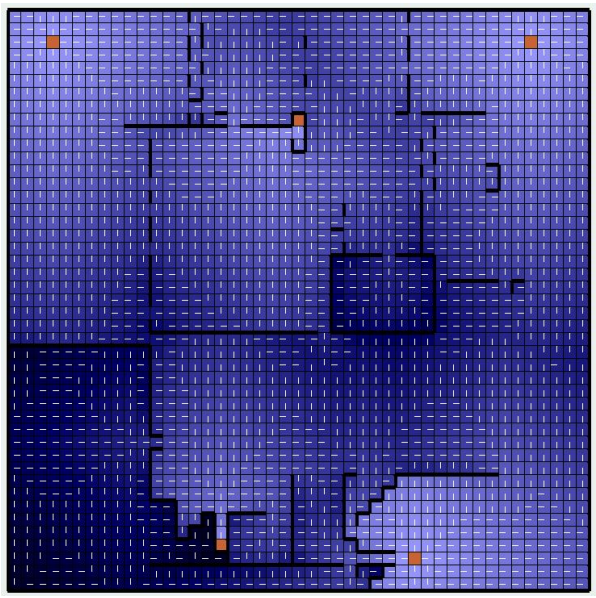
Policy iteration result (penalty for hitting the wall = 100, PJOG = 0.3)



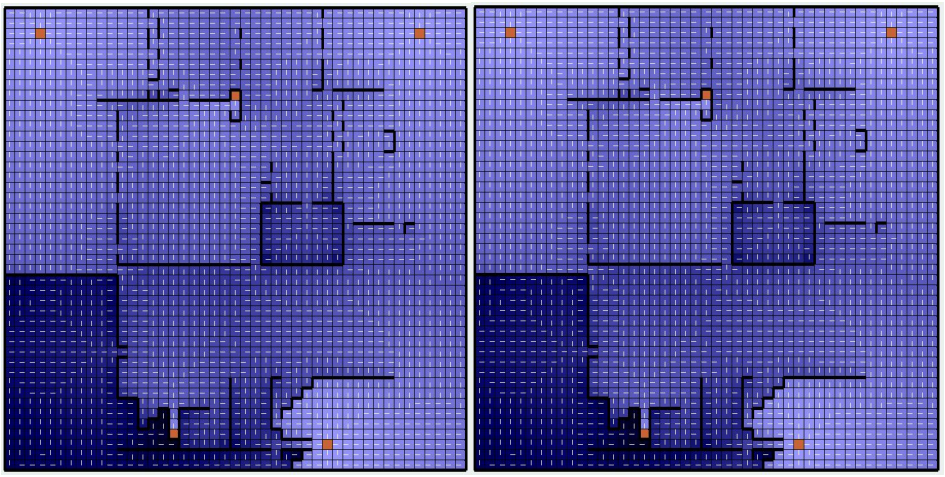
Value iteration result (penalty for hitting the wall = 100, PJOG = 0.1)



Policy iteration result (penalty for hitting the wall = 100, PJOG = 0.3)



3. Discount factor: After change the discount factor from 1 to 0.5, we can see that the values for states that are far away from the starting points decreased. This is because it takes more steps to go to those states from the initial point, and the reward is discounted due to the discount factor. Therefore, the agent is more likely to find the goal state near the starting point. The graph is shown in next page.



Value iteration (left) and Policy iteration (right), discount factor 0.5, PJOG 0.1

4. State Representation: As mentioned in MAZE1 problem. Unfortunately, it is a not a good choice to change the state representation in grid world problems.