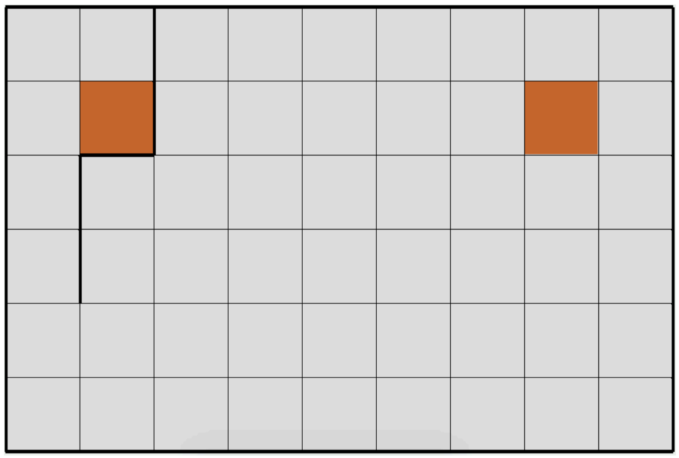
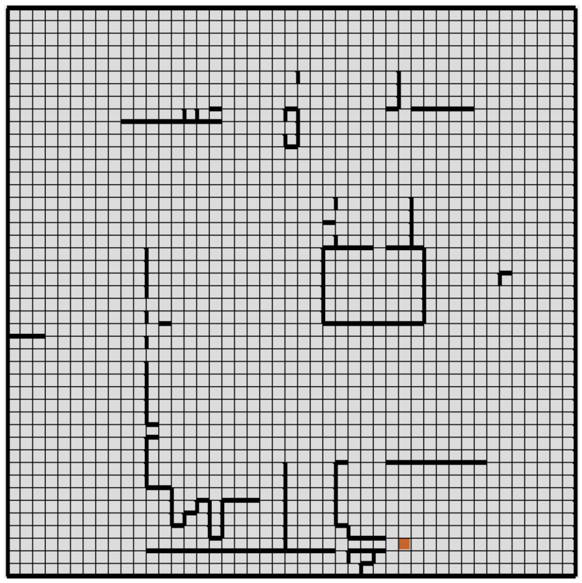
Hardik Sangwan
CS 4641

MDPs and Reinforcement Learning

MDP Problems
I used the following two grid world mazes for an MDP based agent to navigate.



The first maze is a simple 9 x 6 grid with 54 states and 2 goals where 1 goal is completely open while the other is surrounded by walls. It will be interesting to see how the optimal path is chosen for both goals given their two differing conditions.



The second is a considerably more complex 45 x 45 maze with 1 goal, a lot of walls and 2025 states. It will interesting to compare this maze with the previous in terms of iterations required and time taken as states are increased.

Variables and GridWorld Rules – (from RL_sim description)
In the maze-world, every cell representing a state. There are one or more goal cells. There can be a wall present on the boundary between any two adjacent cells. Agent occupies cell completely at any given instant. Agent can take four actions: UP, DOWN, LEFT, RIGHT, which tries to take it to the adjacent state in corresponding direction. It is not possible for agent to cross the walls and it gets penalty if it hits a wall. Also every step taken in maze costs and agent gets penalty equivalent to path cost. Agent get zero reinforcement on reaching any goal state.

PJOG – Takes on a value between 0 and 1 to model Environment Noise. If agent wants to take action UP, then agent take that action with probability (1-PJOG) and it takes one of remaining action with probability PJOG/3. (Varied from .1 - .5 in all experiments)

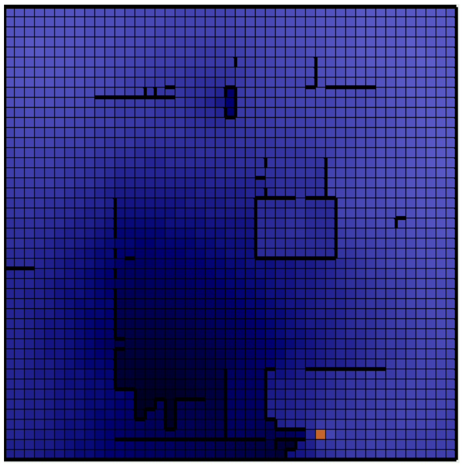
Precision Value – minimum convergence limit (set at a default value of .001 for all experiments since it is already known that larger tolerances decrease required time but also performance while lower tolerances increase time and performance).

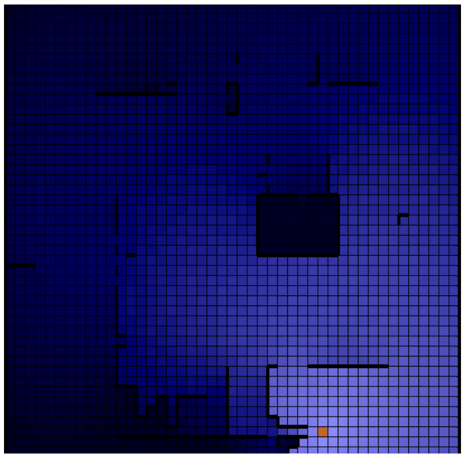
MDPs Definitions, Results and Discussion

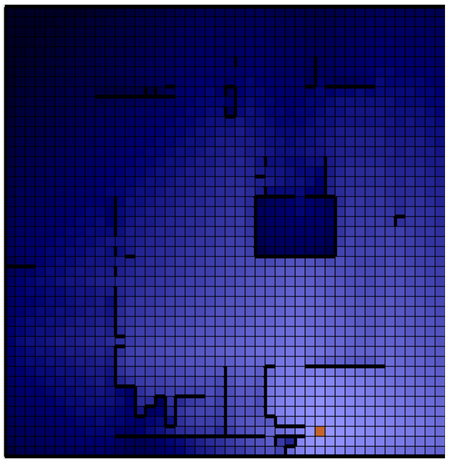
Maze Number	Algorithm	PJOG	Time (ms)	Iterations
1	Value Iteration	.1	1	24
1	Value Iteration	.3	3	48
1	Value Iteration	.5	11	123
1	Policy Iteration	.1	18	5
1	Policy Iteration	.3	14	4
1	Policy Iteration	.5	22	4
1	Policy Iteration	.7	26	4
1	Policy Iteration	.9	9	3
2	Value Iteration	.1	1746	100
2	Value Iteration	.3	2852	164
2	Value Iteration	.5	6502	373
2	Policy Iteration	.1	10288	49
2	Policy Iteration	.3	10305	15
2	Policy Iteration	.5	13248	15
2	Policy Iteration	.7	216363	9
2	Policy Iteration	.9	2049	161

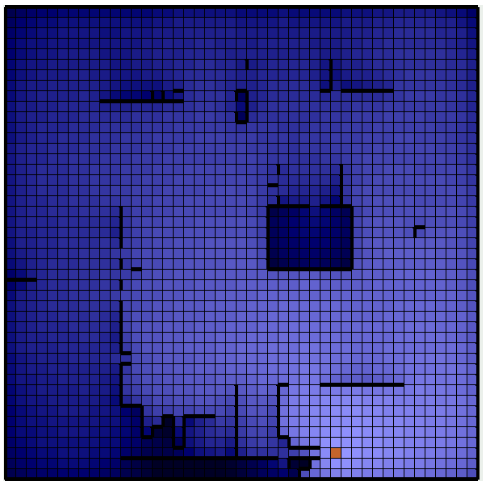
Value Iteration runs until changes in state value between iterations are lower than the set precision.

Policy Iteration runs by evaluating the policy in each iteration and improving it in the next step. Policy is evaluated by calculating the expected infinite discounted reward for all states for the current policy. It terminates for zero policy changes between iterations.


PJOG – 0.9


PJOG – 0.7

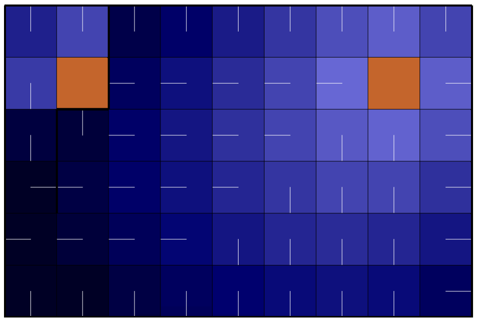

PJOG – 0.5

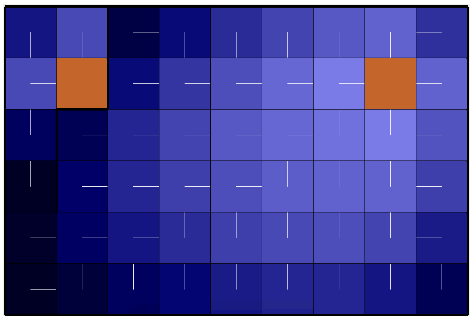

PJOG – 0.1

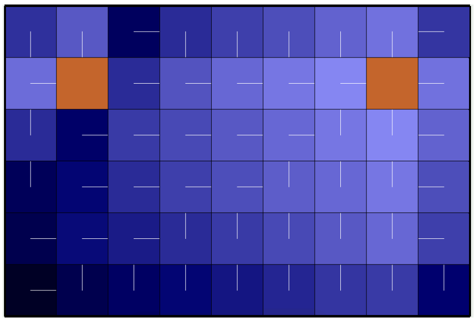
Policy Iteration Maze 2 Grids

Lower PJOG leads the agent much closer to the walls while it tends to be in open spaces for higher PJOG. For Policy Iteration on Maze 2, values close to the mean of .5 have the lowest iterations while iterations are higher on both ends. But in the case of Maze 1, iterations and time both tend to go down as PJOG is increased. This leads to the conclusion that policy evaluation trends depend greatly on the grid. We also see though that number of states directly correlate with time required and number of iterations.

Value Iteration Maze 1 Grids


PJOG – 0.9

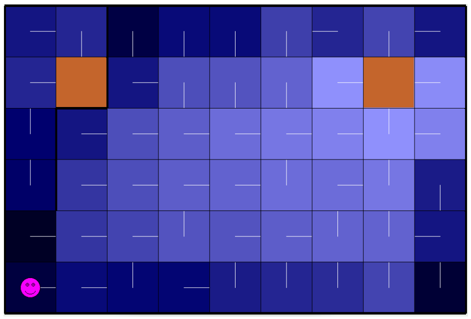

PJOG – 0.5

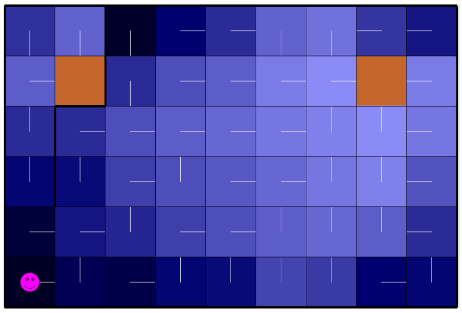

PJOG – 0.1

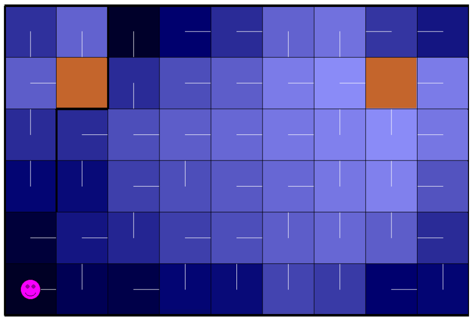
The trends in value iteration are much more straightforward. Increased PJOG leads to increased iterations and therefore increased time. This can also be correlated to the fact that higher PJOG forces the agent to go towards a path it did not choose to go in.

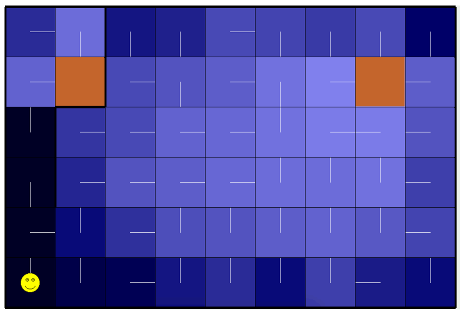
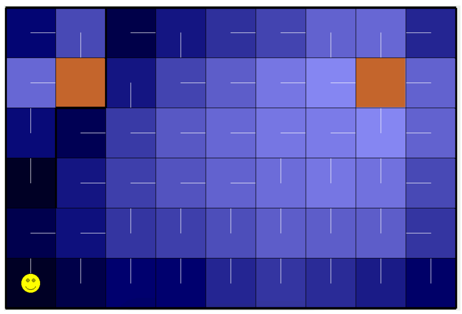
Q Learning Definitions, Results and Discussion

[From RL_sim help manual] **Q Learning** is a model free algorithm that maintains the values for the state action pairs, called Q values, it experiences. The action in each state is chosen according to the ε-greedy policy. It uses a parameter called as learning rate, $\alpha \in [0,1]$ to update the Q values for each state it experiences. As the name signifies it determines how quickly the agent learns from the environment to discover the optimal policy. High learning rate leads to faster improvements but more oscillations between optimal and suboptimal policies because of high sensitivity to reinforcements. The experiments were run using a decaying learning rate because it can take advantage of both a high learning rate which can initially imbibe the dynamics of environment quickly and then a low learning rate that does not oscillate but settles to an optimal policy. Epsilon controls the exploration vs exploitation side of Q Learning where a low epsilon value leads to low exploration. Below are the experiment grids with epsilon, learning rate and life cycle changes (PJOG .3).





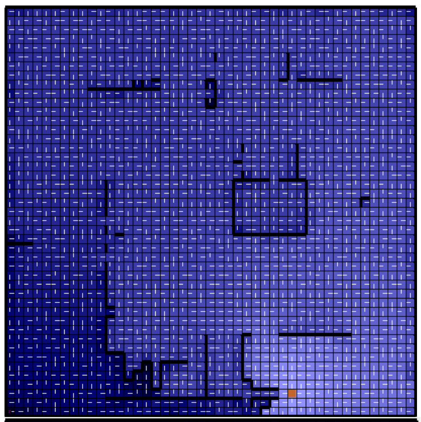
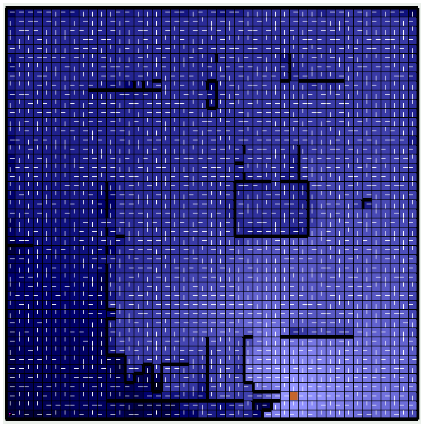
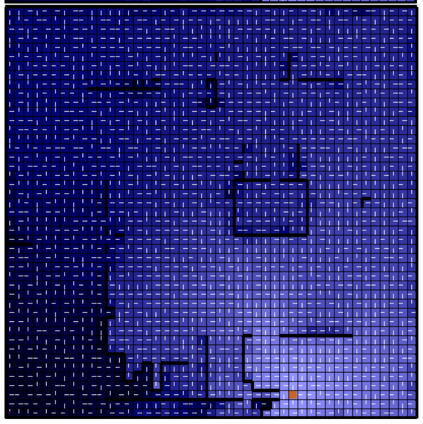


Above - Epsilon, varied from .1 to .9 left to right (learning rate .3, 1000 cycles)

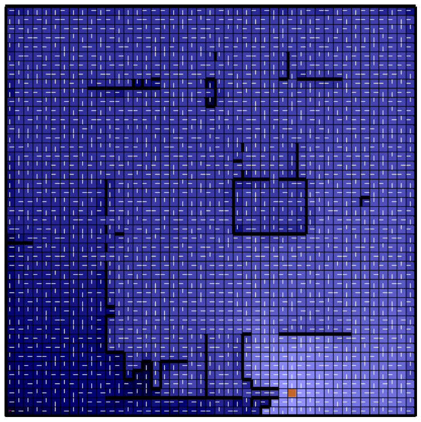
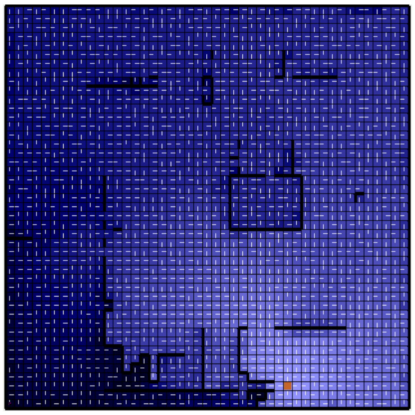
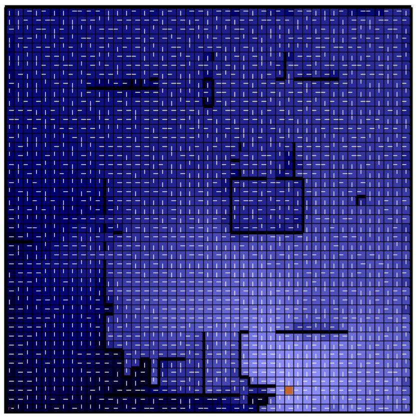
Optimal values are found at lower values with less penalties whereas there are increased rewards and suboptimal results at higher values. Similarly increased penalties are found near walls for higher epsilon for maze 2.

Below – learning rate varied from .2 to .8 left to right (epsilon .1, 1000 cycles)

Again, more penalties around walls for a higher learning rate, probably due to increased oscillations at higher rates, but similar results overall given the decaying learning rate selected.

Below – cycles varied 1000 – 100000 left to right (epsilon .1, learning rate .1)

Increase in cycles having little effect on improving policy as it has probably been selected at the 1000 cycles mark and increased cycles only leading to slightly increased penalties near the bottom walls, the overall pattern otherwise remaining quite similar. Similar results were seen for maze 1 as optimal policy was reached at a low number of cycles and given the low epsilon and learning rate, increased cycles led to only slight deviation from the results already obtained at lower cycles.

Conclusion

In conclusion, Policy iteration takes significantly more time but less iterations as it computes the entire Markov Chain from the start state. Thus policy iteration is ideal for problems where the action space is large because it considerably reduces the action space in lesser number of iterations. Value iteration is ideal for problems where the state space is large as it takes a lot less time for more iterations because it only computes values of neighbor states. Q Learning is the most computationally intensive and requires a lot more changes to its parameters to find the optimal policy but can do so without prior model/space knowledge and once the optimal parameters have been found, the resulting policy is equally good or better than value/policy iteration.