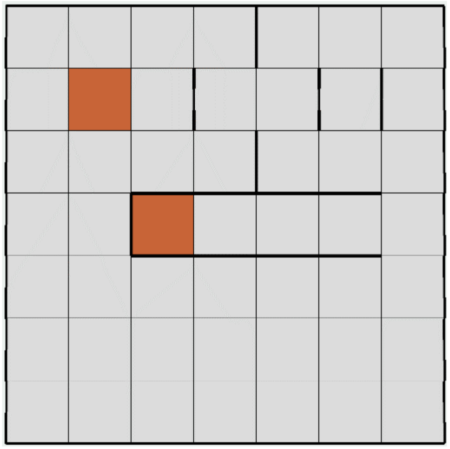
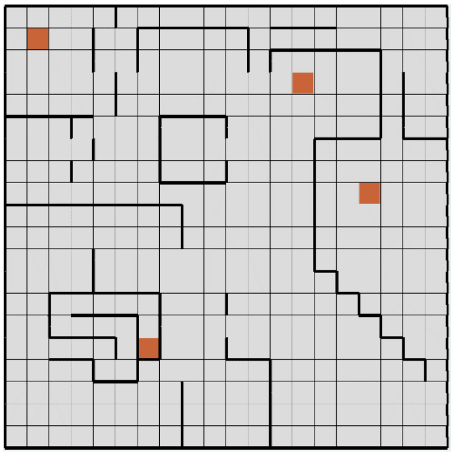
Description of MDPs:





I created two gridworld mazes for my MDPs using RL_sim. For each maze, you receive -1 reward for being in any state that isn't a goal, and you also receive -50 reward for bumping into a wall. Reaching a goal or orange square will end the game. So, the agent will attempt to reach a goal as quickly as possible while avoiding walls. The transition function is determined by the PJOG parameter, which represents the probability that instead of the intended action the agent is sent to a random adjacent state.

The first MDP has a relatively small number of states, as it is a 7 by 7 maze and therefore has 49 states. This problem is interesting because there are two goals; however, one is surrounded by lots of walls and the other is in a much more open space. It will be interesting to see in which states a long path is preferable to a short one.

The second MDP has a much larger number of states, as it is a 20 by 20 maze and therefore has 400 states. Similarly, this MDP has multiple goals, so it will again be interesting to see when longer paths are preferable to shorter ones. Also, the larger number of states makes this problem interesting as we can now compare the two MDPs and see if the number of states affects each algorithms performance.

Value Iteration:

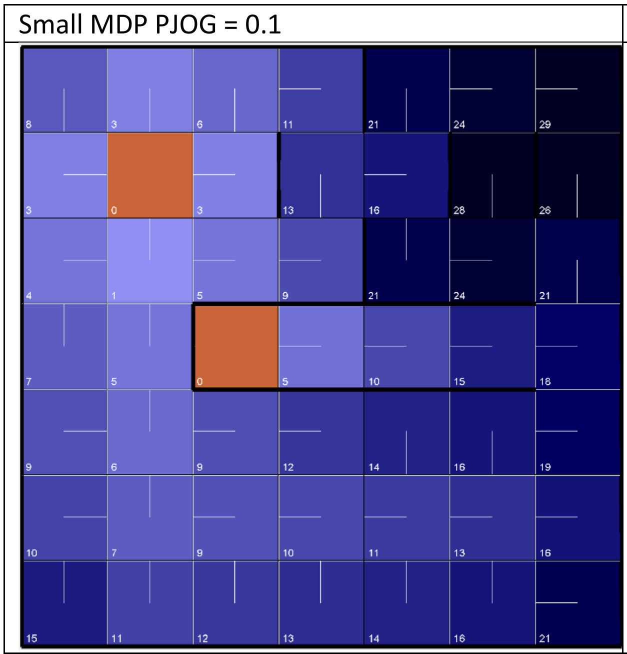
The table below shows how many iterations and how long it took for value iteration to converge to 0.001 precision for different PJOG values.

MDP	PJOG	Iterations	Time
Small	0.05	20	2ms
Small	0.1	25	6ms
Small	0.3	57	11ms
Large	0.05	42	112ms
Large	0.1	51	143ms
Large	0.3	102	278ms

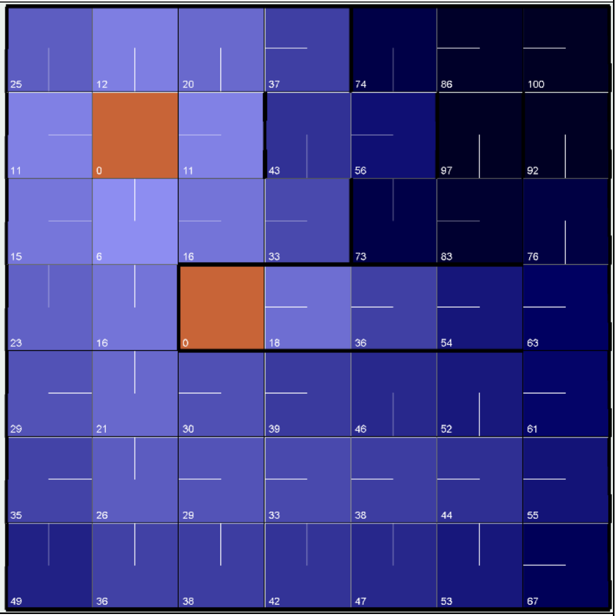
We can see for higher PJOG values, the algorithm needed more time and iterations to converge. This makes sense as the agent is more likely to perform unintended actions for higher PJOG values, and one would expect this noise to cause the convergence to be slower.

Since each iteration is $O(S^2A)$ (where S is the number of states), we see that the larger MDP with about 8 times as many states takes much longer than the smaller MDP to converge.

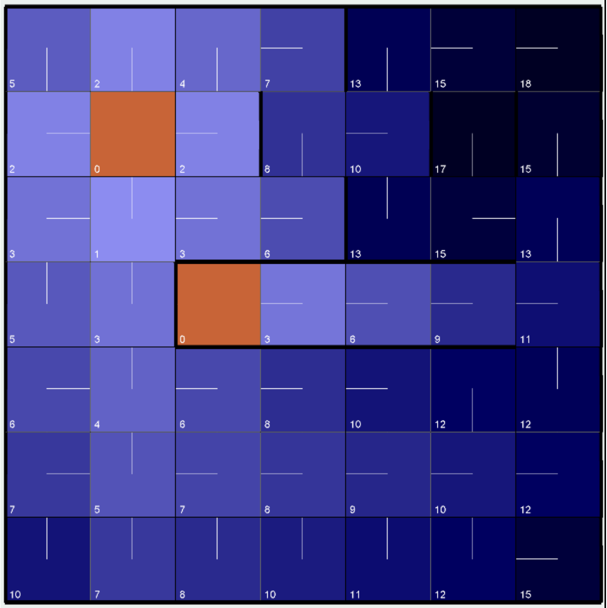
Small MDP PJOG = 0.1



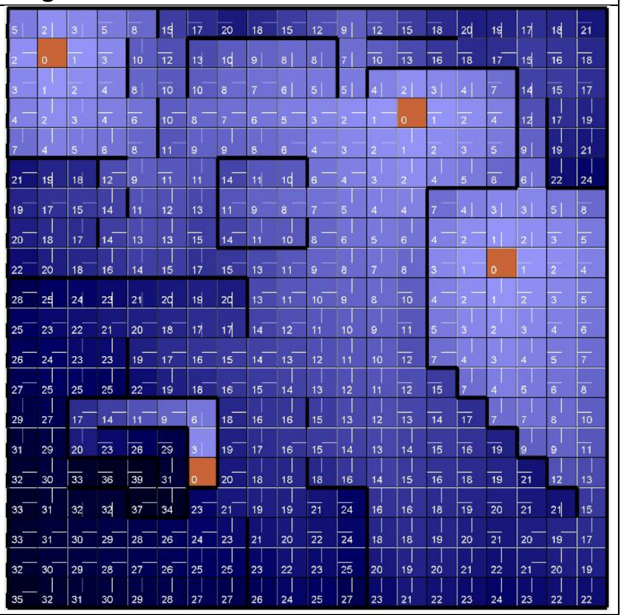
Small MDP PJOG = 0.3



Small MDP PJOG = 0.05

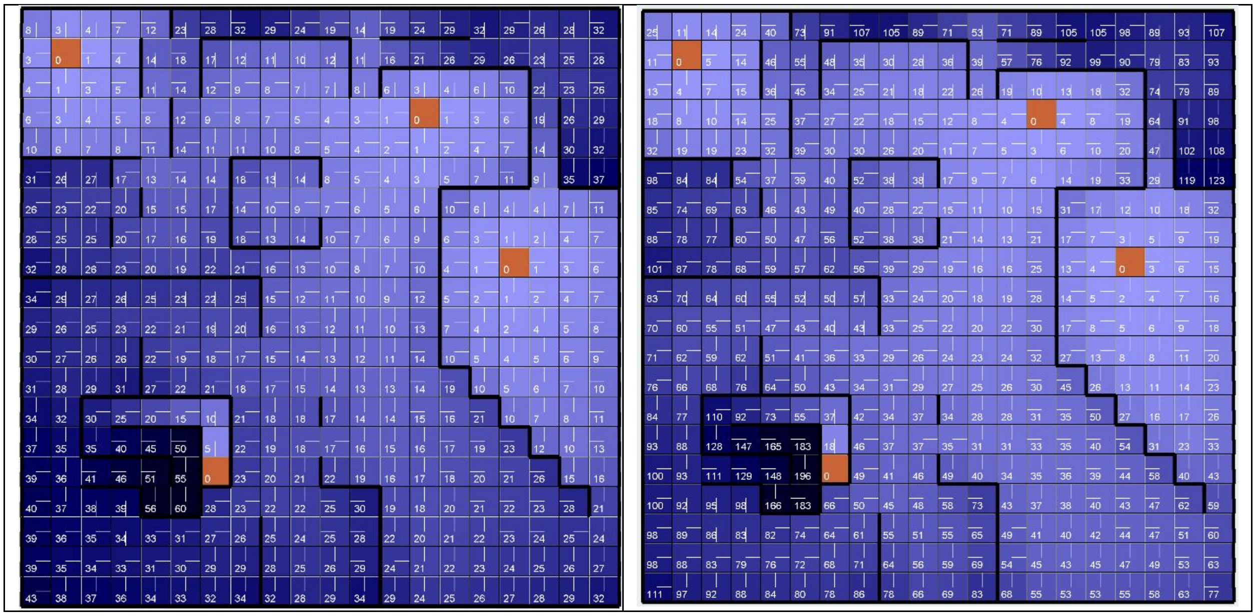


Large MDP PJOG = 0.05

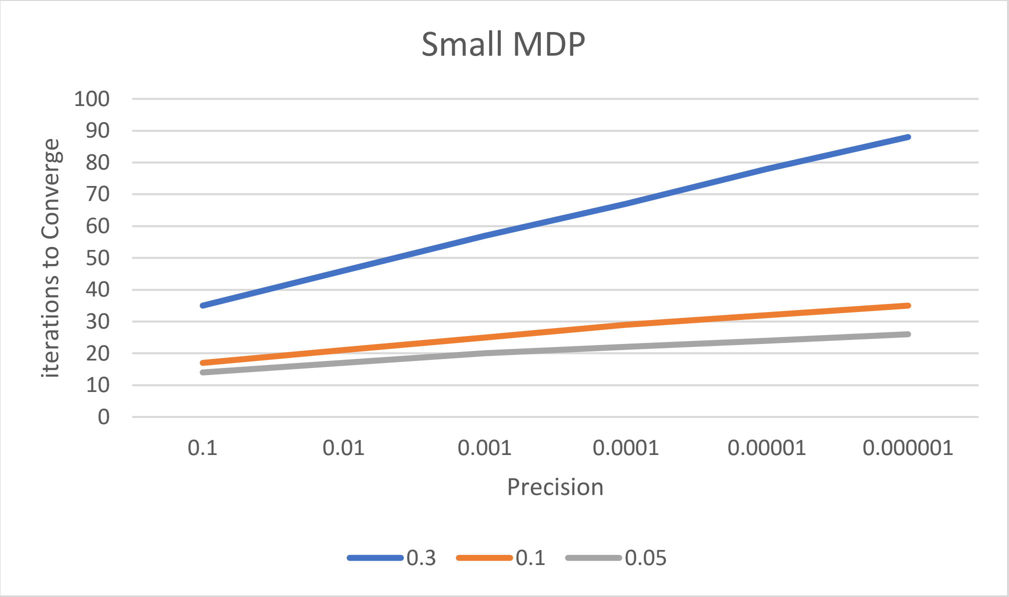


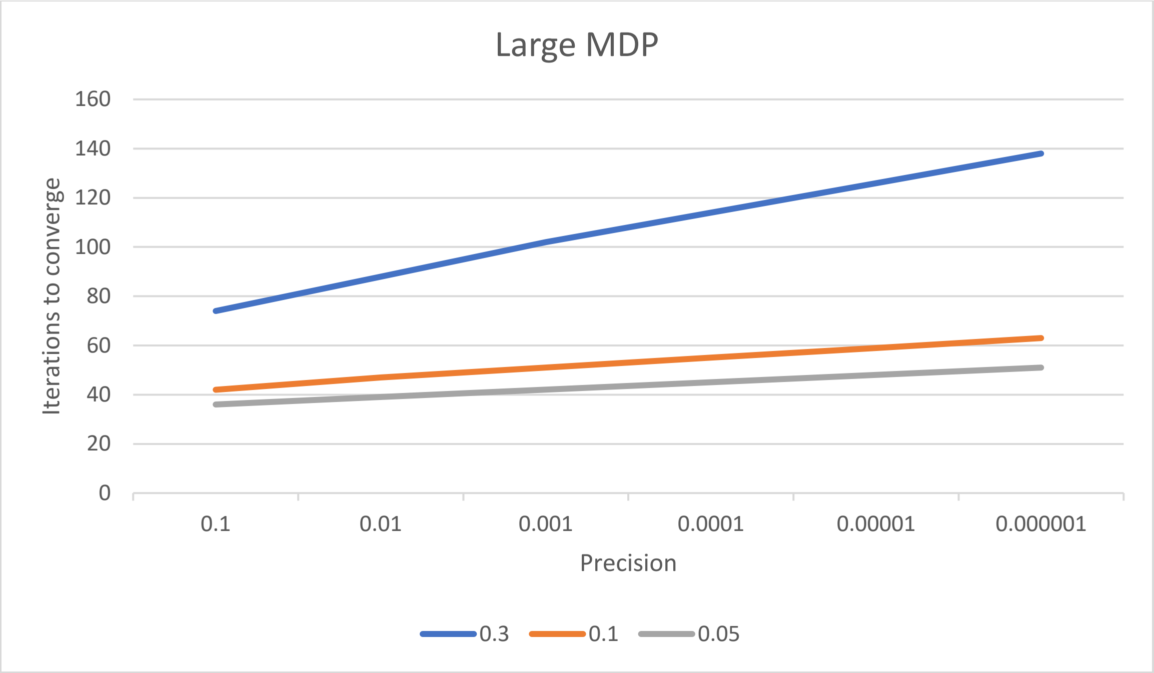
Large MDP PJOG = 0.1

Large MDP PJOG = 0.3



Above is the resulting policy from value iteration for different PJOG values. For the small MDP, we can see in the bottom right side, the policies differ between 0.3, 0.1 PJOG and 0.05 PJOG. The agent chooses a shorter path when the PJOG is 0.05 because its probability of hitting the walls is much lower than it is for PJOG = 0.1 or 0.3. For the large MDP, we can see similar effects as for the small MDP. For example, in the bottom right corner, for PJOG = 0.3 the agent chooses to go left and up to the farther goal, and for PJOG = 0.05 and 0.1, the agent chooses to go up to the closer goal, because it is no longer so weary of passing through the walls. As the PJOG approaches 0, the agent will just take the shortest path to a goal since its actions happen with absolute certainty. And as PJOG increases (but not too close to 1), the agent will take a path that avoids walls over the shortest path.





The above graphs show the relationship between precision and the number of iterations needed to converge for each MDP, we can see they are logarithmically related. This is called linear convergeance which is where the number of iterations required is linear in the number of bits that need to be precise. The linear convergeance factor which is related to the slope of the graph clearly depends on the PJOG and the number of states. Interestingly, there were no differences between the policies for different precision values. This is probably because the values converge after the optimal policy is reached for values lower than 0.1.

Policy Iteration:

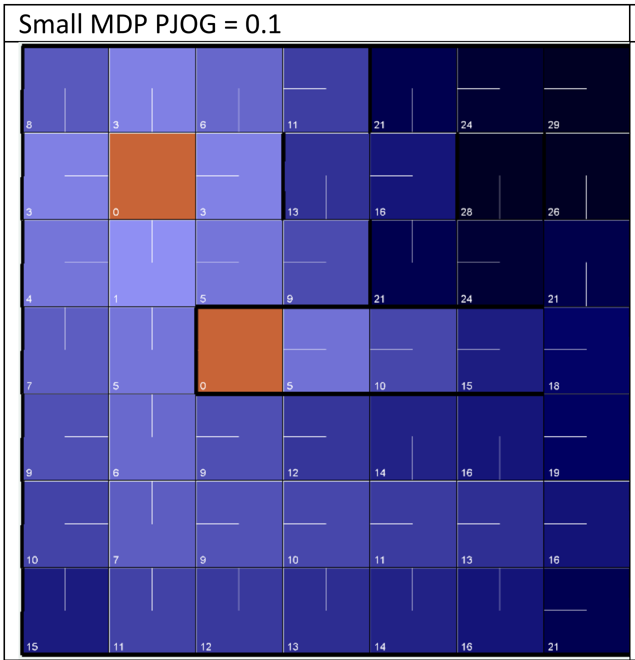
Below is the number of iterations and time for the policy iteration to converge with max iterations set to 500 and max value set to 100,000 (a large number a chose so as not to limit the values) and precision of 0.001.

MDP	PJOG	Iterations	Time
Small	0.05	4	9ms
Small	0.1	5	10ms
Small	0.3	4	11ms
Large	0.05	8	938ms
Large	0.1	8	717ms
Large	0.3	9	847ms

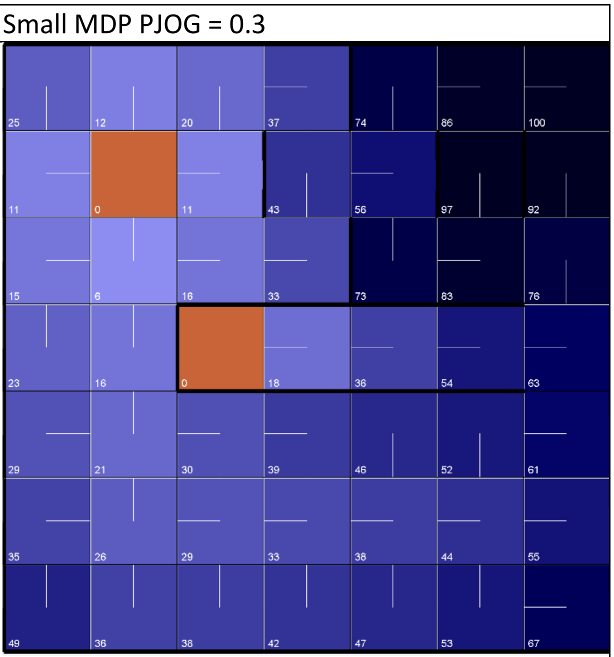
We can see the numer of states had an affect on both the iterations and time. The time required for convergeance increased by a much larger factor than the iterations though. There didn't seem to be much correlation between the PJOG and the amount of time and iterations required for convergeance.

Small MDP PJOG = 0.1

Small MDP PJOG = 0.3



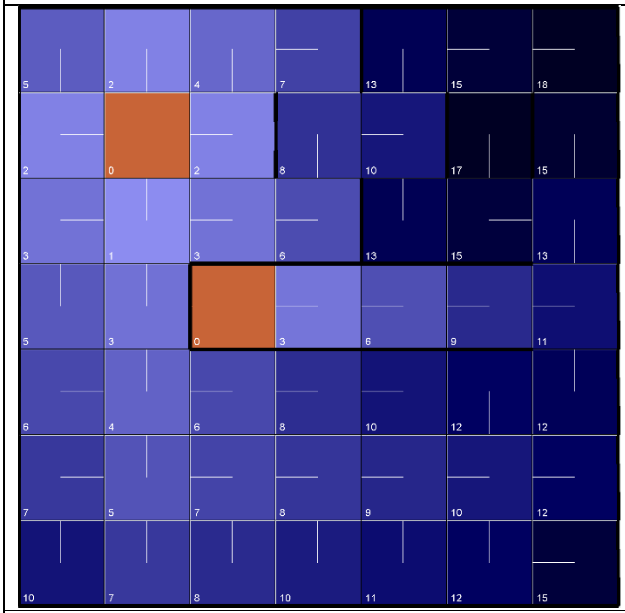
8	3	6	11	21	24	29
3	0	3	13	16	28	26
4	1	5	9	21	24	21
7	5	0	5	10	15	18
9	6	9	12	14	16	19
10	7	9	10	11	13	16
15	11	12	13	14	16	21



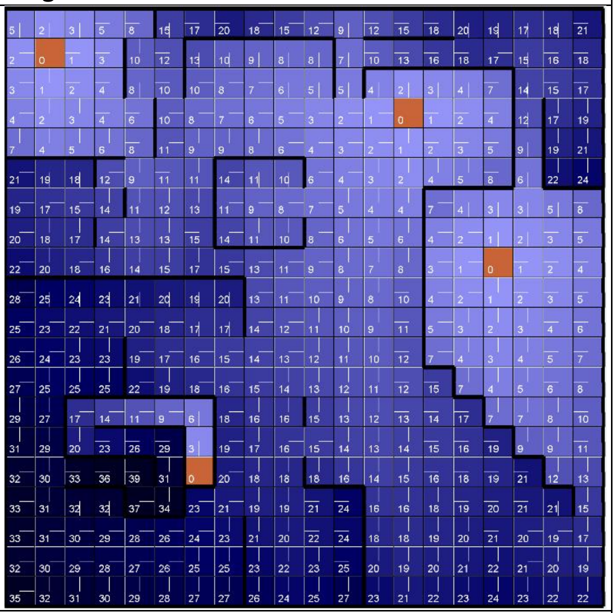
25	12	20	37	74	86	100
11	0	11	43	56	97	92
15	6	16	33	73	83	76
23	16	0	18	36	54	63
29	21	30	39	46	52	61
35	26	29	33	38	44	55
49	36	38	42	47	53	67

Small MDP PJOG = 0.05

Large MDP PJOG = 0.05



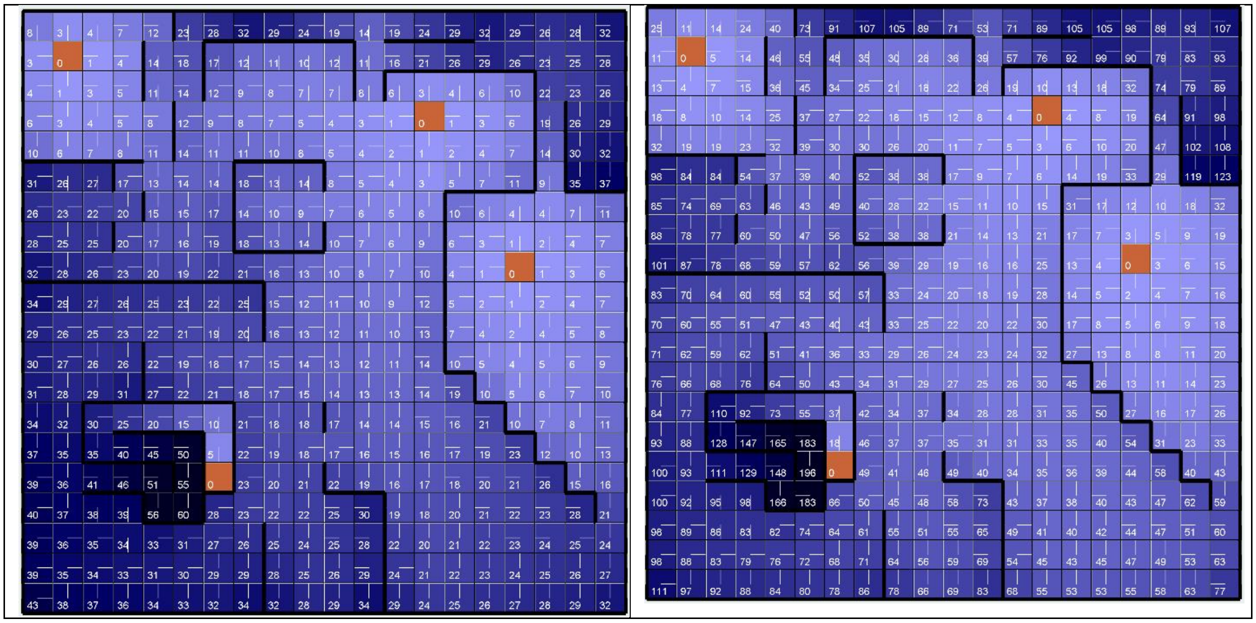
5	2	4	7	13	15	18
2	0	2	8	10	17	15
3	1	3	6	13	15	13
5	3	0	3	6	9	11
6	4	6	8	10	12	12
7	5	7	8	9	10	12
10	7	8	10	11	12	15



5	2	3	5	8	15	17	20	18	15	12	9	12	15	18	20	19	17	18	21
2	0	1	3	10	12	13	10	9	8	8	7	10	13	16	18	17	15	16	18
3	1	2	4	8	10	10	8	7	6	5	5	4	2	3	4	7	14	15	17
4	2	3	4	6	10	8	7	6	5	3	2	1	0	1	2	4	12	17	19
7	4	5	6	8	11	9	9	8	6	4	3	2	1	2	3	5	9	19	21
21	19	18	12	9	11	11	14	11	10	6	4	3	2	4	5	8	6	22	24
19	17	15	14	11	12	13	11	9	8	7	5	4	4	7	4	3	3	5	8
20	18	17	14	13	13	15	14	11	10	8	6	5	6	4	2	1	2	3	5
22	20	18	16	14	15	17	15	13	11	9	8	7	8	3	1	0	1	2	4
28	25	24	23	21	20	19	20	13	11	10	9	8	10	4	2	1	2	3	5
25	23	22	21	20	18	17	17	14	12	11	10	9	11	5	3	2	3	4	6
26	24	23	23	19	17	16	15	14	13	12	11	10	12	7	4	3	4	5	7
27	25	25	25	22	19	16	16	15	14	13	12	11	12	15	7	4	5	6	8
29	27	17	14	11	9	6	18	16	16	15	13	12	13	14	17	7	7	8	10
31	29	20	23	26	29	3	19	17	16	15	14	13	14	15	16	19	9	9	11
32	30	33	36	39	31	0	20	18	18	18	16	14	15	16	18	19	21	12	13
33	31	32	32	37	34	23	21	19	19	21	24	16	16	18	19	20	21	21	15
33	31	30	29	28	26	24	23	21	20	22	24	18	18	19	20	21	20	19	17
32	30	29	28	27	26	25	25	23	22	23	25	20	19	20	21	22	21	20	19
35	32	31	30	29	28	27	27	26	24	25	27	23	21	22	23	24	23	22	22

Large MDP PJOG = 0.1

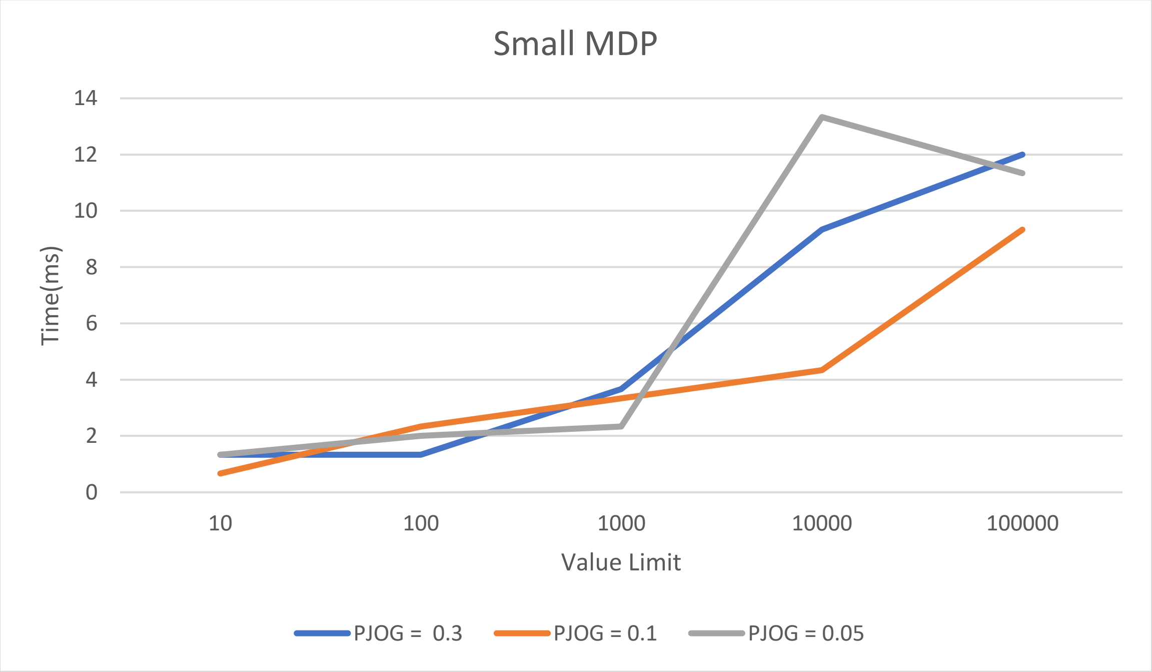
Large MDP PJOG = 0.3

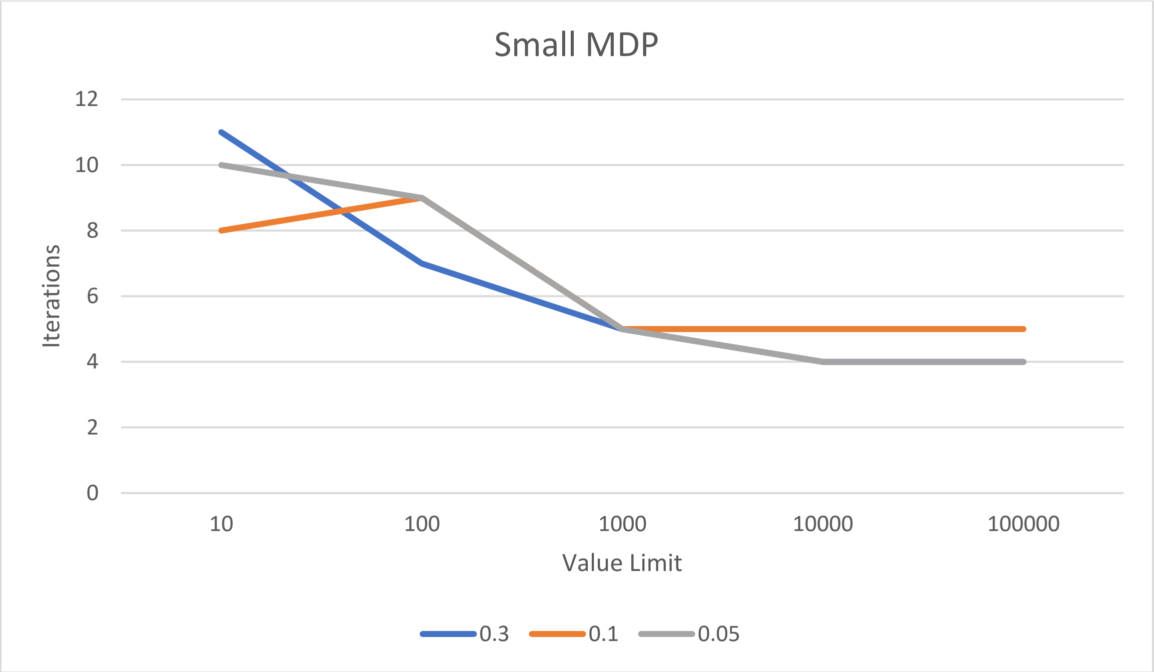


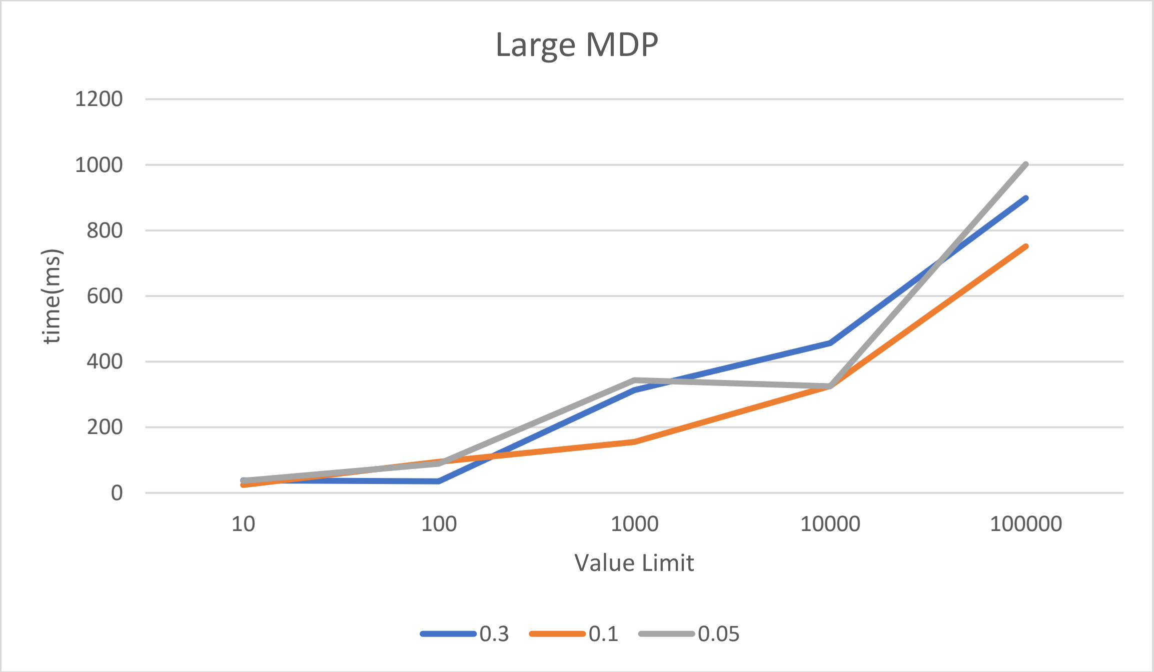
Above are the policies for each different PJOG value. These are the exact same policies that value iteration converged to. This is expected as both methods converge to the optimal policy.

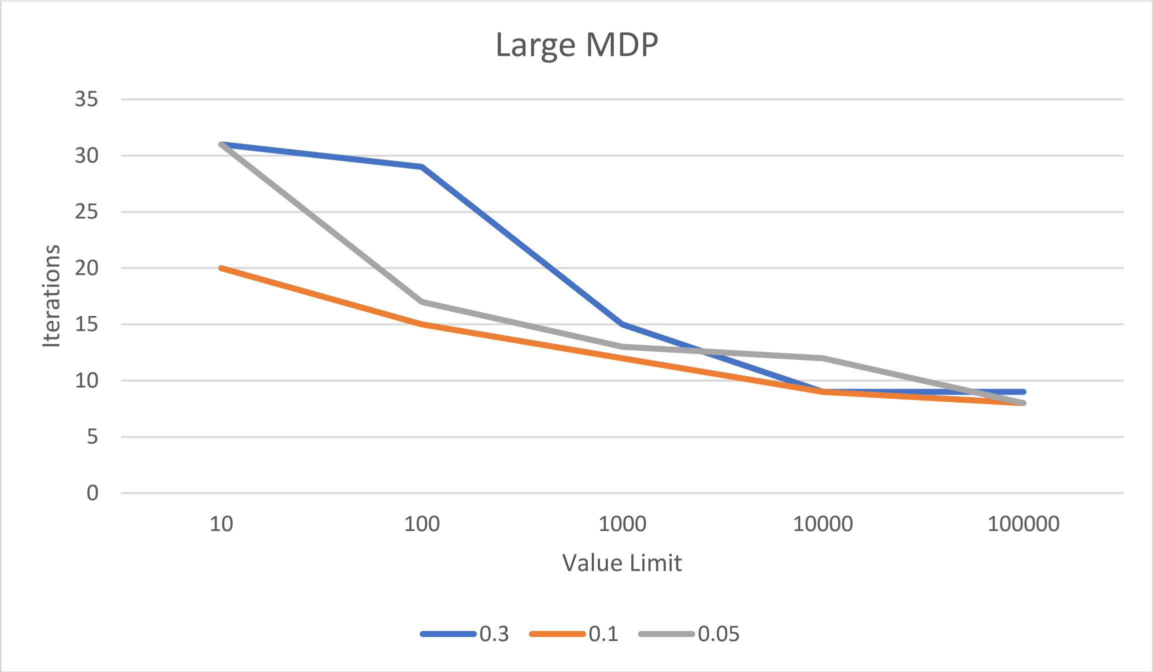
Below, I decided to experiment with input values to see the effect on the iterations and time taken to converge. I first changed the precision which had no effect at all. So, I chose to change the Value Limit paramter, and I believe it is unnecessary to change both the value limit parameter and the max iterations parameter, since the iterations will effectively be limited by the value limit parameter when the limit is exceeded.

The belowe experiments were run with precision of 0.001 and max iteration value of 500.

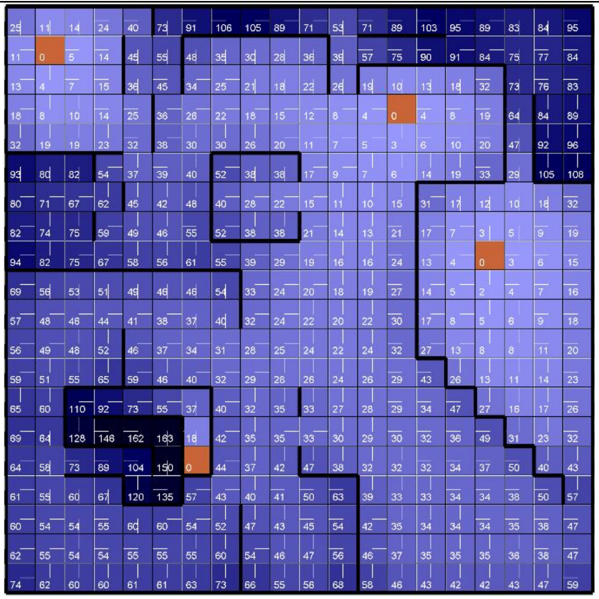
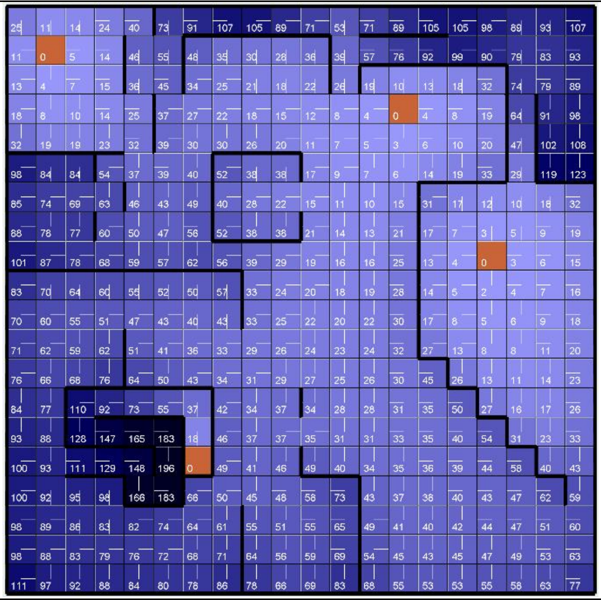



Small MDP
12
10
8
6
4
2
0
Iterations
10
100
1000
10000
100000
Value Limit
0.3
0.1
0.05


Large MDP
1200
1000
800
600
400
200
0
time(ms)
10
100
1000
10000
100000
Value Limit
0.3
0.1
0.05



We can see for lower value limits the time requireed for convergeance decreased dramatically, while the iterations needed increased. The value limit will effectively stop the computations when they become too time consuming, however this will make the policy update worse and require more iterations to converge. There were only minimal differences in the policies for each different value limit.

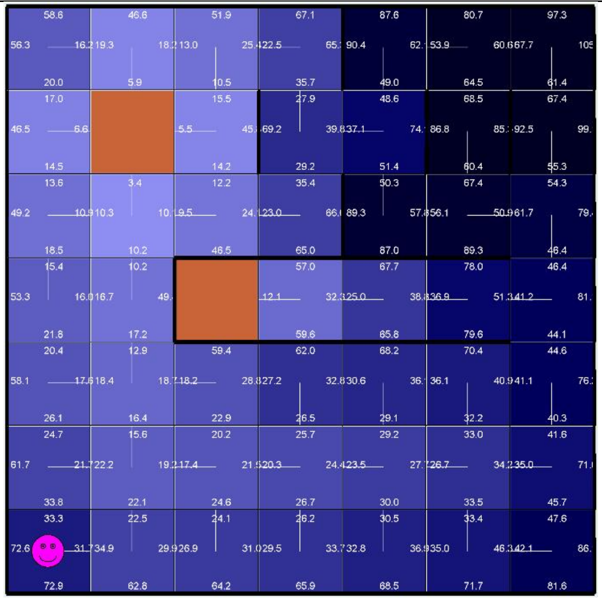
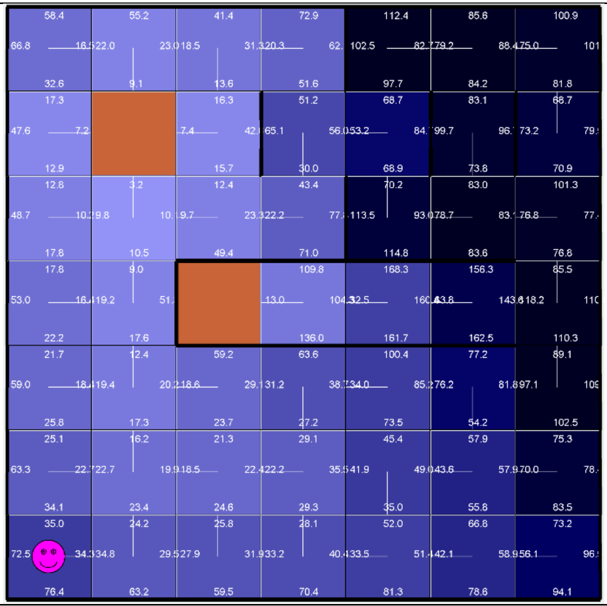
Value limit = 10	Value limit = 100,000
	

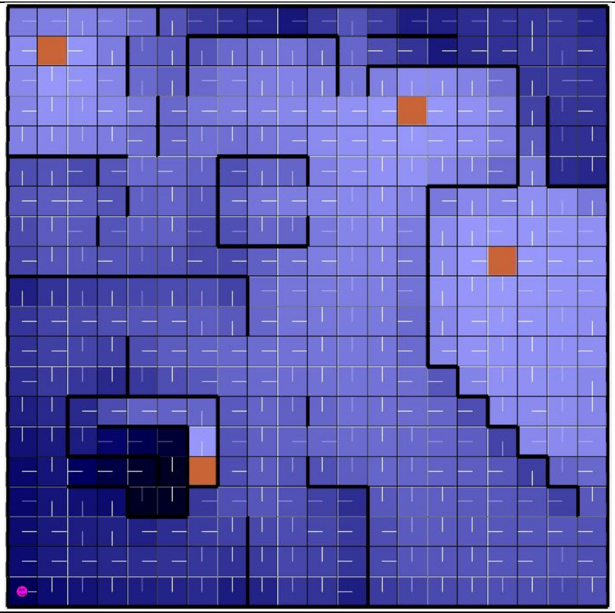
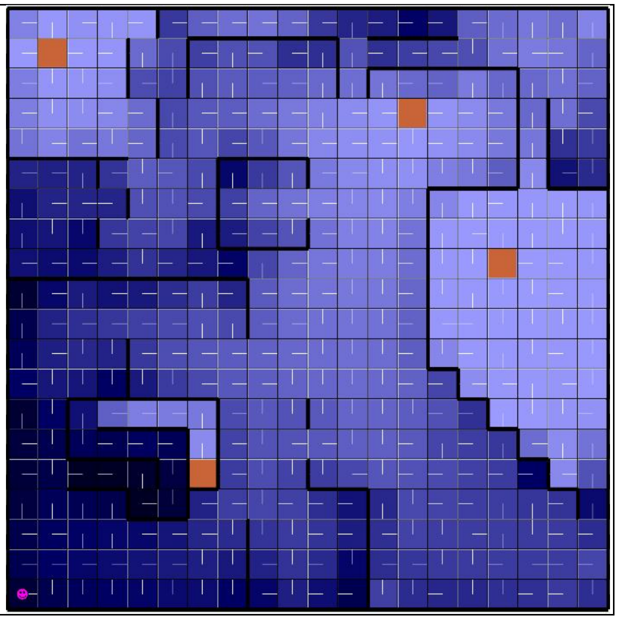
The above were run with PJOG = 0.3. There are only minimal differences between these two policies, however the one with a value limit of 10 took 42ms, and the one with a value limit of 100,000 took 847ms. So by limiting the values, we can get a almost optimal policy for almost one twentieth of the time.

In comparison with the value iteration data in the previous section, we see that value iteration took many more iterations to converge. This makes sense since value iteration waits until the values converge whereas policy iteration stops after the policy converges which might and often will happen long before the values converge. However, despite the lower number of iterations the policy iteration takes longer to converge, this is because each iteration takes longer since it must evaluate the entire markov chain from any state, whereas value iteration just computes the values of the neighboring states in one iteration. However, if change the value limit for policy iteration, we can get almost optimal policies for even faster than value iteration, 42ms on the Large MDP with PJOG =0.3, compared to 278ms for value iteration.

Q-Learning:

I decided to use RL_sim's Q Learning algorithm to solve each MDP. I performed the below experiments with PJOG = 0.3, Precision = 0.001, and a decaying learning rate of 0.7.

Small MDP Epsilon = 0.5, 100,000 cycles	Small MDP Epsilon = 0.1, 100,000 cycles
	
Large MDP Epsilon = 0.5, 1,000,000 cycles	Large MDP Epsilon = 0.1, 1,000,000 cycles

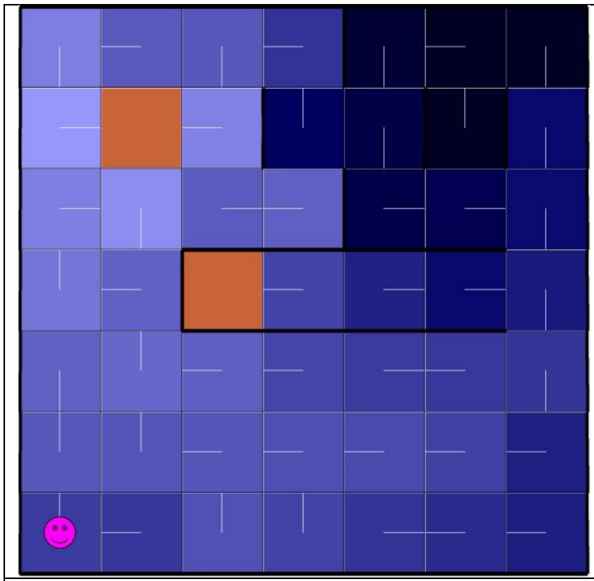
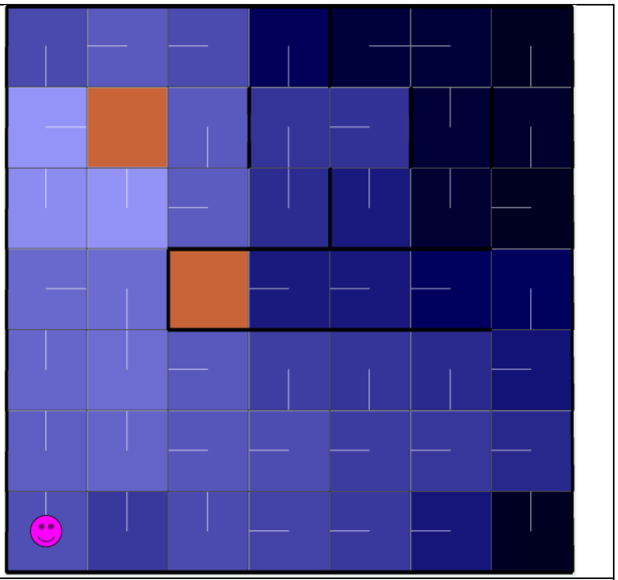
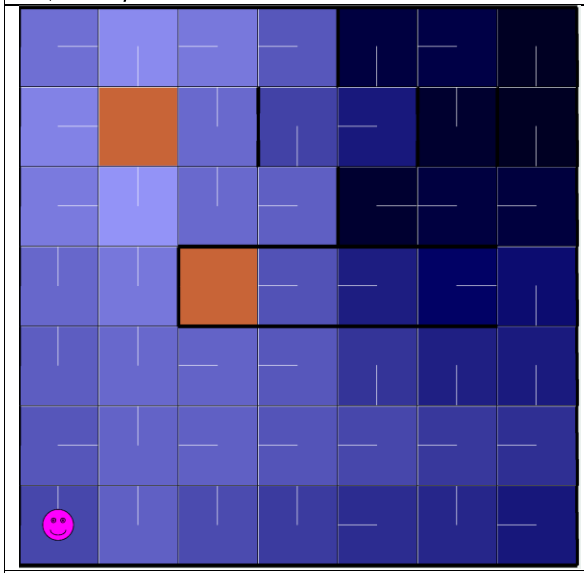
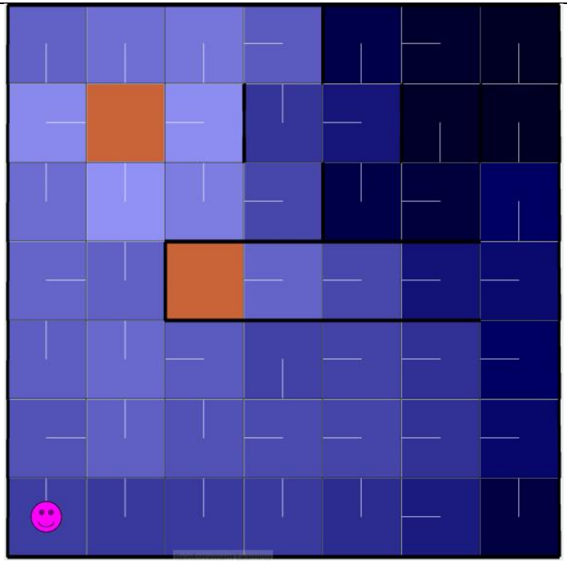
	

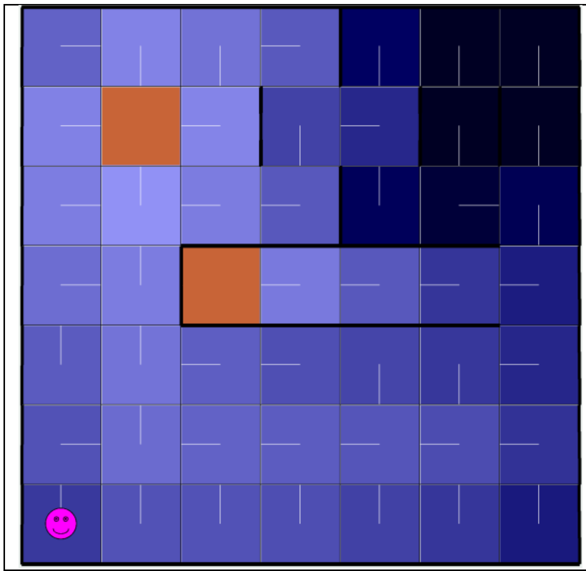
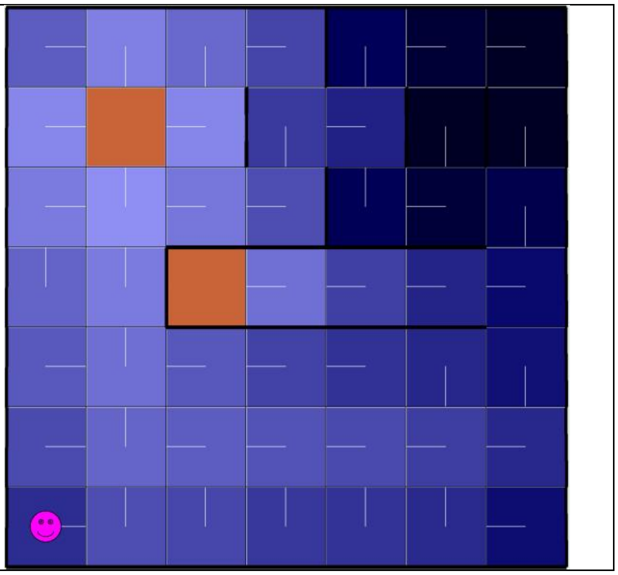
Compared to Policy iteration and value iteration, these policies are suboptimal. Also, Q learning took much longer, taking almost 2 minutes for the large MDP with epsilon = 0.5 and 1,000,000 cycles, which does resemble the policies found in policy and value iteration. The main advantage of Q Learning being the algorithm does not use prior knowledge of the state space and transition model.

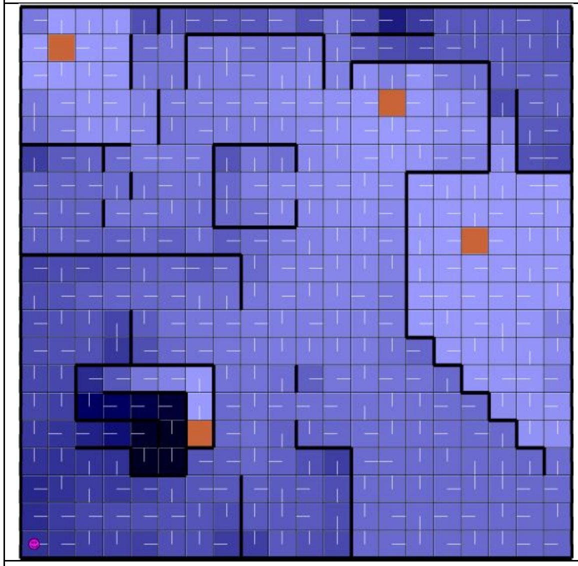
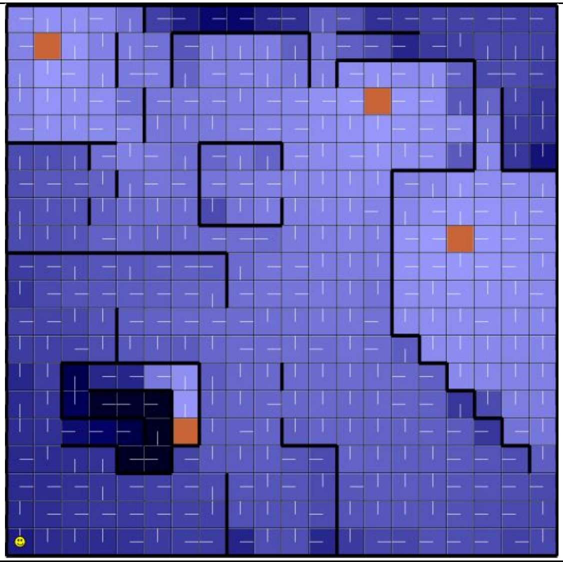
We can see the larger epsilon value produced policies much more similar to the policies found more likely to explore each state a large of number of times, hence this is closer to the goal of visiting each state an infinite number of times. However, the larger epsilon value took much more time than the smaller one. The epsilon determines the tradeoff between exploration and exploitation, with the higher epsilon valued agents choosing to explore more and the lower epsilon value agents heading quickly to the goal or exploiting current knowledge.

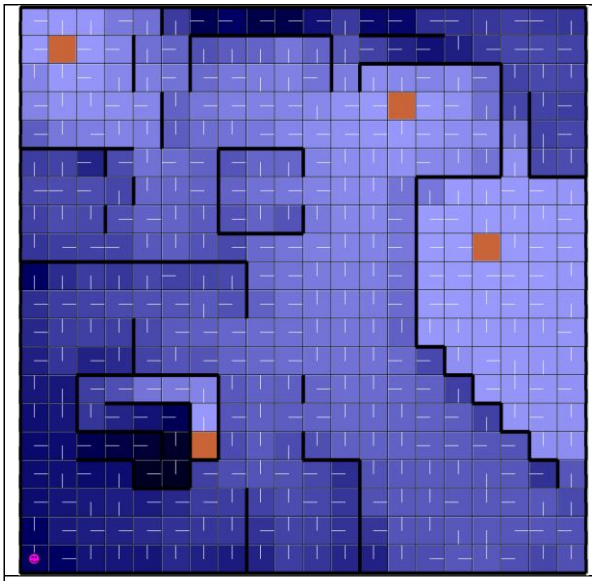
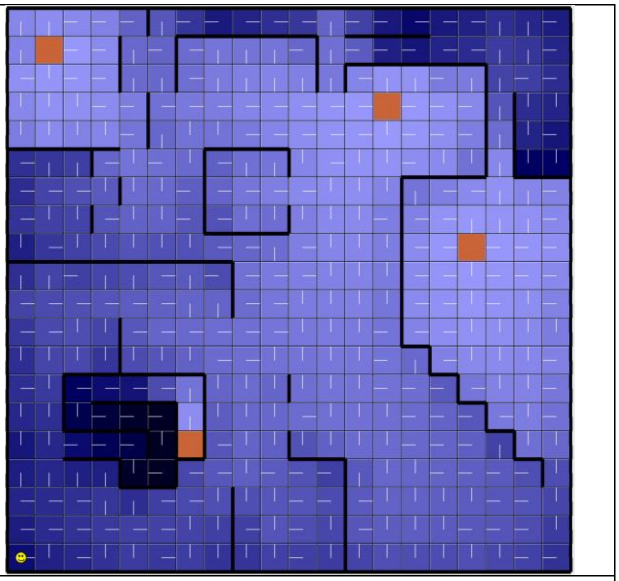
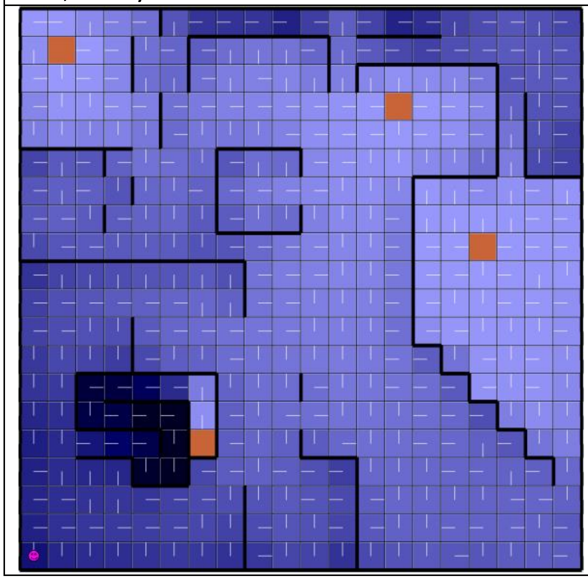
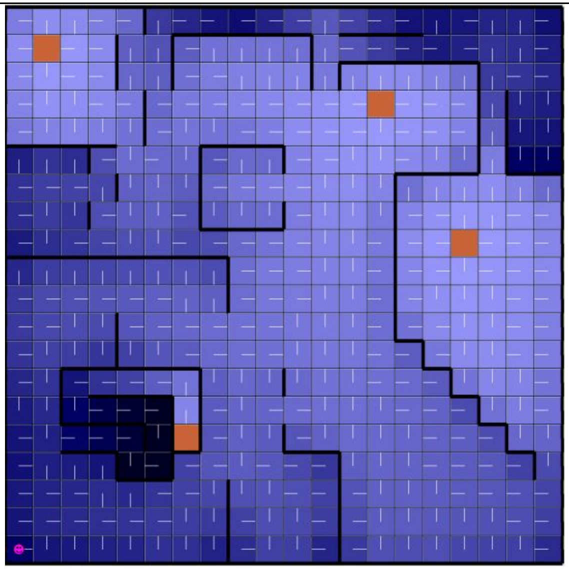
I experimented with the number of cycles run, we can see the results below. These were run with epsilon of 0.5, 0.99 and the same parameters as above.

Epsilon = 0.5	Epsilon 0.99
1,000 cycles	1,000 cycles

	
10,000 cycles	10,000 cycles
	
100,000 cycles	100,000 cycles

Epsilon = 0.5	Epsilon = 0.99
1,000 cycles	1,000 cycles
	
10,000 cycles	10,000 cycles

	
100,000 cycles	100,000 cycles
	

As above, we can again see that the higher epsilon value 0.99 in this case produces policies much closer to the ones found during value iteration and policy iteration. We can also see how the 0.99 epsilon agent finds shorter paths to goals than the 0.5 epsilon agent even at only 1,000 cycles, because of its higher chance to explore.

As the number of cycles increases the differences between the different values of epsilon seem to be less significant. This is because as the number of cycles increase each state will be visited more time regardless of the value of epsilon, hence the policy is closer to the optimal. We can also see that states farther from a goal seem to be more effected by the epsilon value. For instance, in the large MDP after 100,000 cycles, the main differences occur in the places like the bottom right corner, which is far from any goal. The 0.5 epsilon one has more dissimilarities from the optimal policy than the .99 epsilon one; however, around any of the four goals the policies are almost identical.