David Li

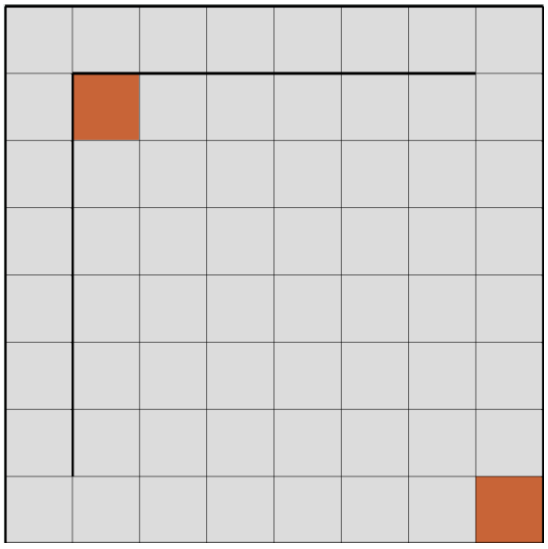
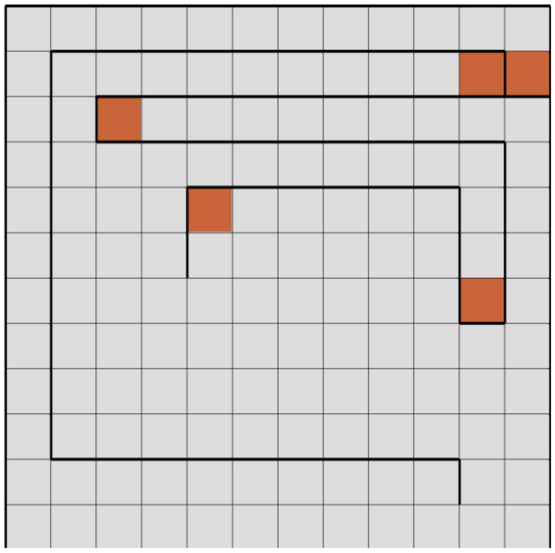
Cs 4641, C. Isbell

April 14, 2019

Markov Decision Process

Overview: As we move to MDP and reinforcement learning, now we try to create models that when given an environment, tries to find the optimal decision and pathing based on the consequences of performing different actions in different states. The algorithms we will be looking at today will include policy iteration, value iteration, and Q-learning, and we will look at how many iterations they take to converge, their real-life runtimes, and the values they produce for each state.

Problems: We will be looking at two different maps created in grid world, one simpler and smaller, while another much more complex and larger in size, allowing comparisons in terms of number of states as well as how obstacles affect the results. While we only needed two problems, I decided to double that by analyzing how having and not having penalties running into walls affect the results, which is allows real life applications in terms of speeding up processes and determining how much randomness to inject when a misstep is penalized.

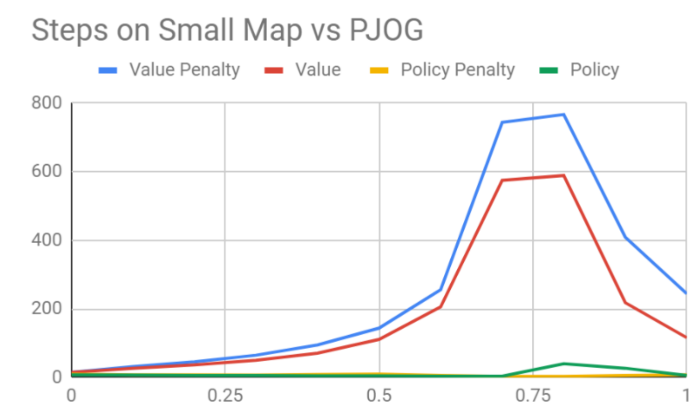



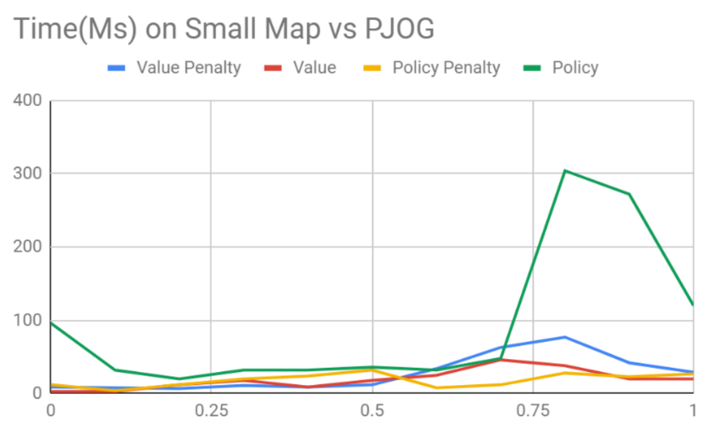
The first maze is relatively simple, with two goal states and 64 states in total. The second one has 144 states and 5 goal states. The first one offers a dilemma at the bottom left and top right corner, where leaving the confines of the way, both goal states are equally distanced, which we shall see how going towards the one in a corner of the wall or without wall will be preferred, especially with wall penalization. The second map has several goal states hidden away in deep hallways, and we shall see how difficult it is to reach them and how the walls of the hallway affects the outcome as well.

Value Iteration vs Policy Iteration

Value iteration gives a utility function to every state, which is determined by looking at its neighboring states and summing up the rewards, while taking into a discount value based on how many steps away it is and finds the maximum of that to set as the value of the state. Every iteration, the value is updated, and indirectly, the policy of the given state is updated as well. It will repeatedly do this until the change in value of the state is below a threshold, thus converging.

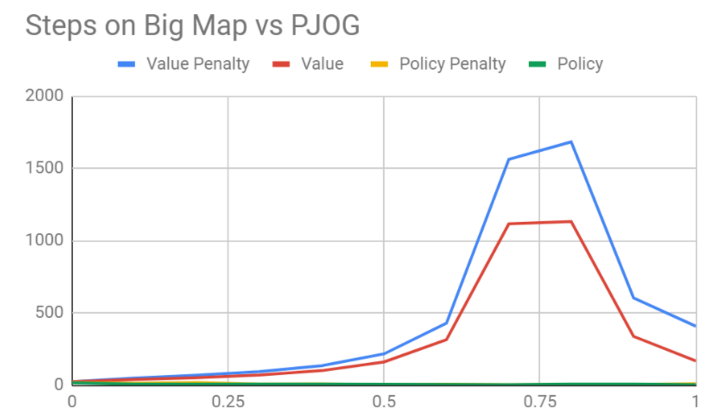
Meanwhile, policy iteration does not wait for the value function of a state to converge, just for the policy to converge, as once the action is decided, it does not matter how accurate the value is, juts that in the given state what you do next. We will calculate the utility several times with fixed policy, and based on the evaluation, a new policy is chosen.

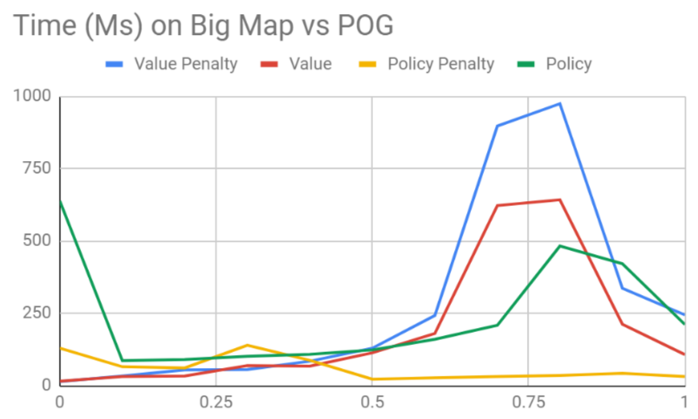




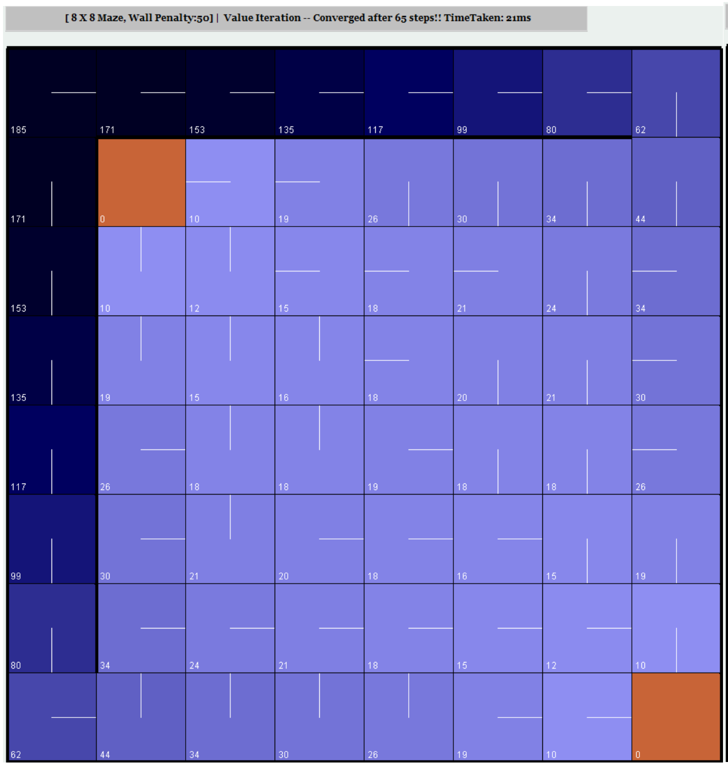
While on the small map, we can see that value iteration took more iterations to converge than policy iteration. This was expected because the point of policy iteration is to decide upon a policy once it is decided to be optimal, which more often than not will occur before the value of each state is converged upon. As the randomness is increased, the amount of iterations taken to converge also increased, especially around .7 to .8, as the algorithm might have optimal policies that the agent does not take more often than not, and if the map allows for specific preffered routes or single most optimal options, than poor results will occur. This is the case where there are two clear goals on the map, and to reach either would be to traverse past the walls and go straight for either one of them, as opposed to going in any other directions. It is to be noted that policy iteration suffered less with randomness, most likely due to the fact that once the action or poliucy is decided, whether it reaches the next optimal state is less important than the action saying it will.

In terms of run time, all the algorithms took around the same except policy without penalities. It may be that because of the symmetry of the map, the algorithm had a difficult time converging upon a policy for some of the states as many are equally good and when one is decided, the rest is based on that single decision in the corner, and when it changes the rest of the neighbors must change too, which the initial state's policy can change at any point since it has two equally optimal policies, causing for a lot of recomputation per iteration.

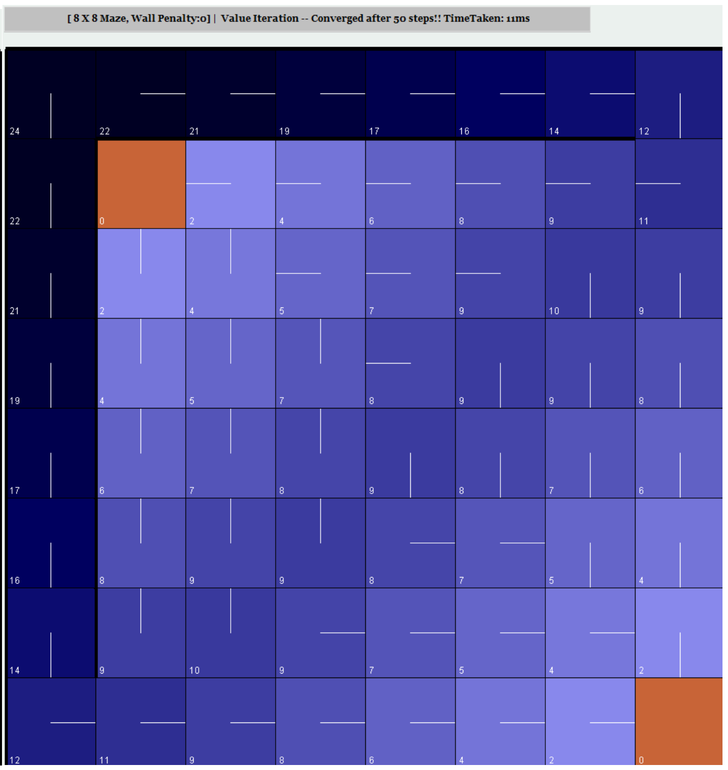




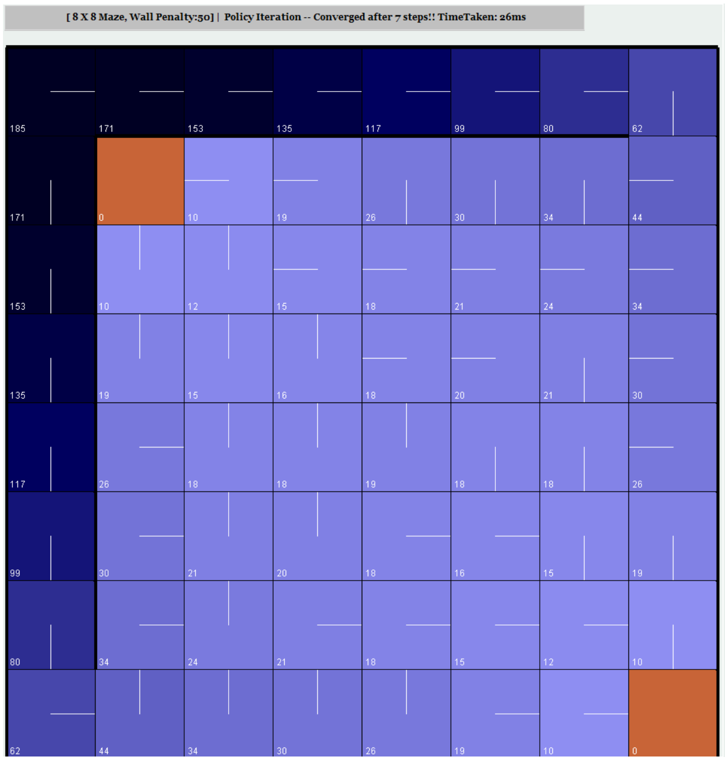
On the bigger map, a similar trend follows, where value iteration lags due to policy iteration converging much earlier, but each line follows its general shape. It is to be noted that the iterations are about doubled from the smaller map for value iteration, which is expected as there are about double the amount of states, which each iteration must take into account all of the following states and their values, while this is less evident in policy iteration as it concerns itself more with the next action concerning just its neighbors. Run time wise again the numbers are much higher than the smaller map, the peak being doubled, but this time every trend besides policy with penalty peaked more around .75. Where given a single optimal path of 4 options, while it is wrong 75% of the time, it becomes much more difficult to converge, but when it becomes completely random, it performs better as it increases the chance of peaking the specific action from 6.25% to 25%. One notable trend is that Value iteration ran slower with penalties while policy ran faster with penalties. It can be because value iteration must recompute another value amongst its utilities as the wall is more than just 0 but -50 and have to be taken into account as an extra neighboring state with possibilities. With policy iteration, having penalties pushes towards a faster decision as very often, in a hallway two of the policies or actions are ruled out almost instantaneously by the algorithm.


[8 X 8 Maze, Wall Penalty:50] | Value Iteration -- Converged after 65 steps!! TimeTaken: 21ms

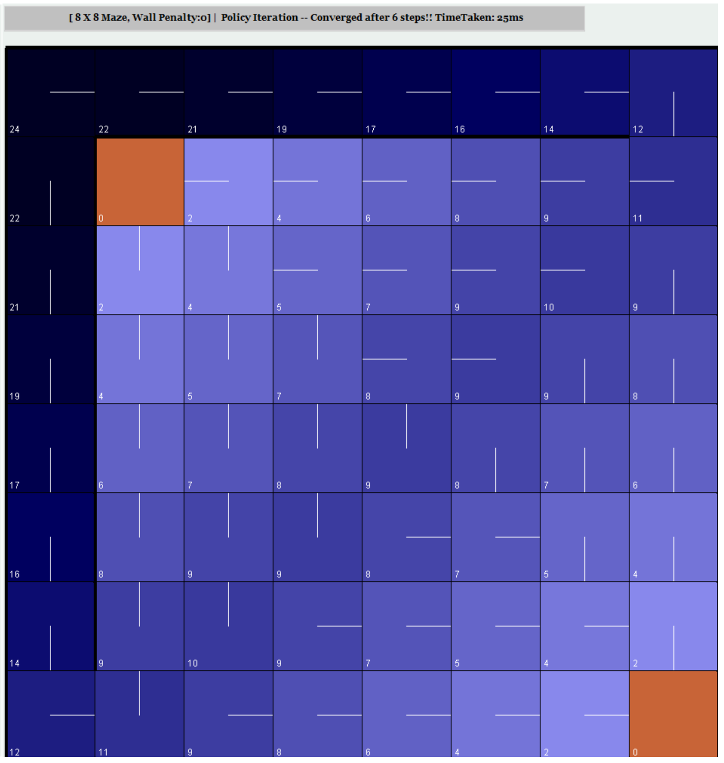
185	171	153	135	117	99	80	62
171	0	10	19	26	30	34	44
153	10	12	15	18	21	24	34
135	19	15	16	18	20	21	30
117	26	18	18	19	18	18	26
99	30	21	20	18	16	15	19
80	34	24	21	18	15	12	10
62	44	34	30	26	19	10	0


[8 X 8 Maze, Wall Penalty:0] | Value Iteration -- Converged after 50 steps!! TimeTaken: 11ms

24	22	21	19	17	16	14	12
22	0	2	4	6	8	9	11
21	2	4	5	7	9	10	9
19	4	5	7	8	9	9	8
17	6	7	8	9	8	7	6
16	8	9	9	8	7	5	4
14	9	10	9	7	5	4	2
12	11	9	8	6	4	2	0


[8 X 8 Maze, Wall Penalty:50] | Policy Iteration -- Converged after 7 steps!! TimeTaken: 26ms

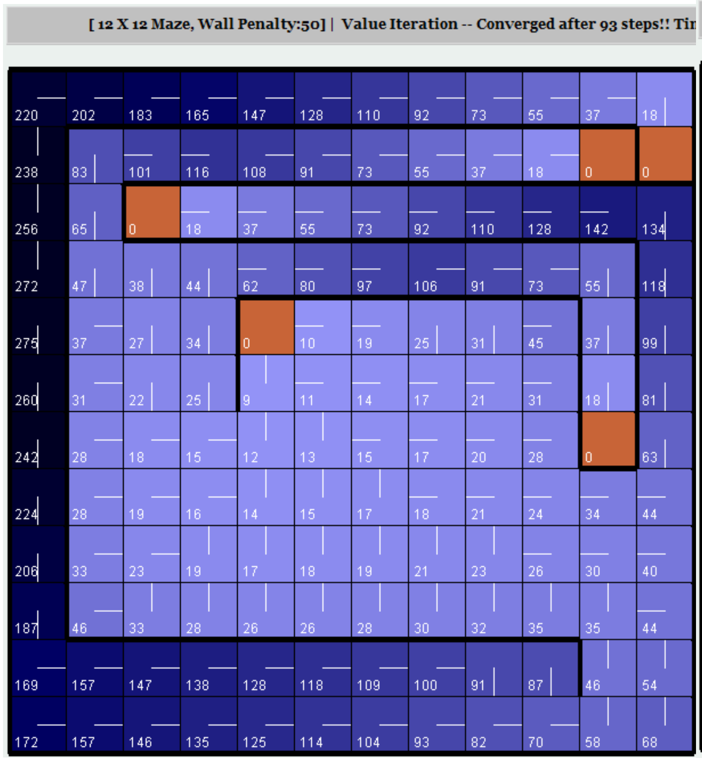
185	171	153	135	117	99	80	62
171	0	10	19	26	30	34	44
153	10	12	15	18	21	24	34
135	19	15	16	18	20	21	30
117	26	18	18	19	18	18	26
99	30	21	20	18	16	15	19
80	34	24	21	18	15	12	10
62	44	34	30	26	19	10	0


[8 X 8 Maze, Wall Penalty:0] | Policy Iteration -- Converged after 6 steps!! TimeTaken: 25ms

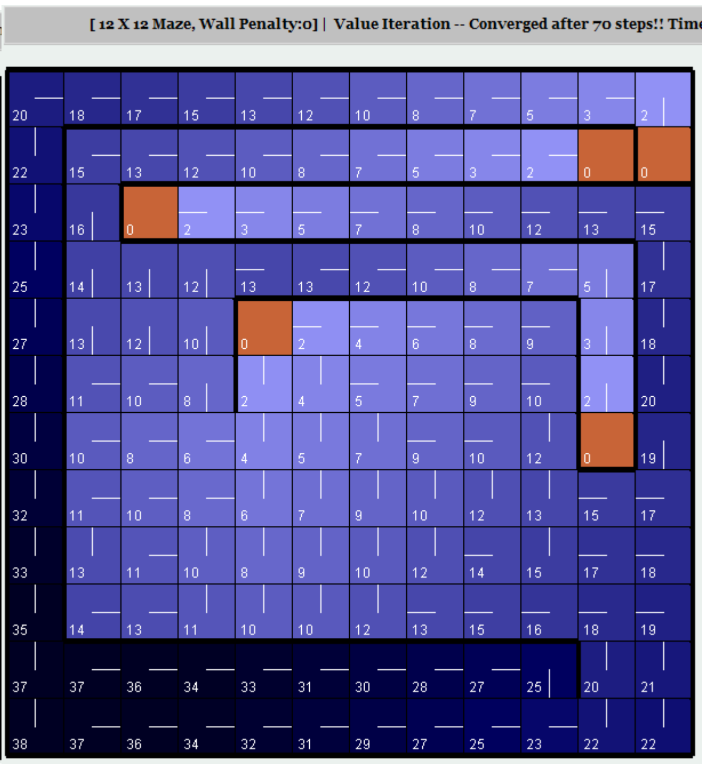
24	22	21	19	17	16	14	12
22	0	2	4	6	8	9	11
21	2	4	5	7	9	10	9
19	4	5	7	8	9	9	8
17	6	7	8	9	8	7	6
16	8	9	9	8	7	5	4
14	9	10	9	7	5	4	2
12	11	9	8	6	4	2	0

Now we look at each of the state values and utilities ran at .3 PJOG. This algorithm prefers a lower score and the Bellman Equation update took the minimum instead of max, for clarification. The top two are value iteration and left two are with penalties. Between the two, there are virtually no difference when using policy or value iteration for the state values, but policy iteration took 10% of the steps value iteration took, as stated before because it can converge on a policy before converging on the state value. This is interesting, as policy iteration still arrived at similar state values, despite that not being the focus. It may also be that the map is smaller, and the state value is more straight forward to calculate.

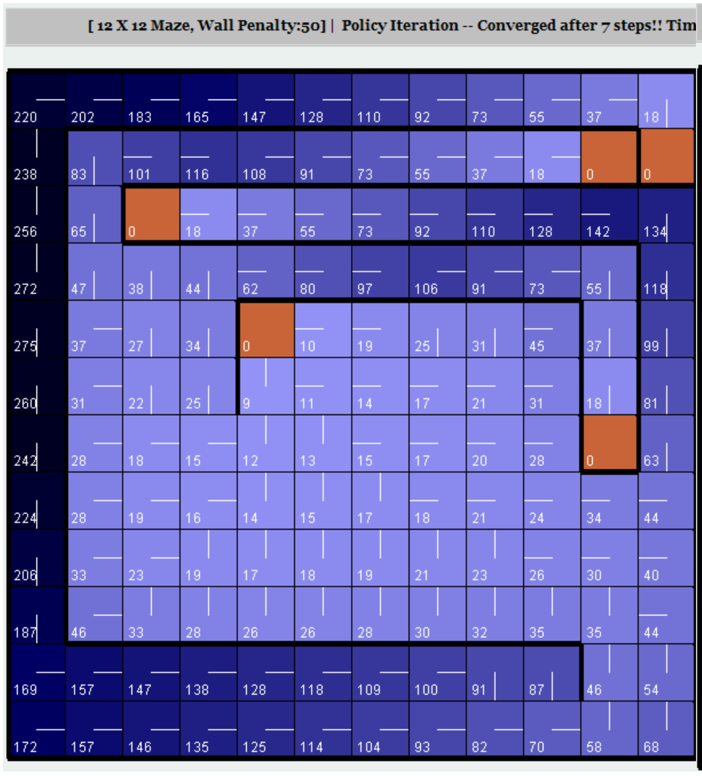
In terms of penalties, having the penalty around greatly hindered the state value as making a wrong decision will cause a lost of 50 points, and because of this it impacted those values following it thus the entire map had worse values, but more peaked increases near the corridor on the side. And as expected, the symmetry made it so the values are identical across the diagonal. In terms of direction, both the algorithm returned nearly identical paths for both types of mazes, np penalty preferred the goal between the walls while penalties preferred the goal with walls.


[12 X 12 Maze, Wall Penalty:50] | Value Iteration -- Converged after 93 steps!! Tir

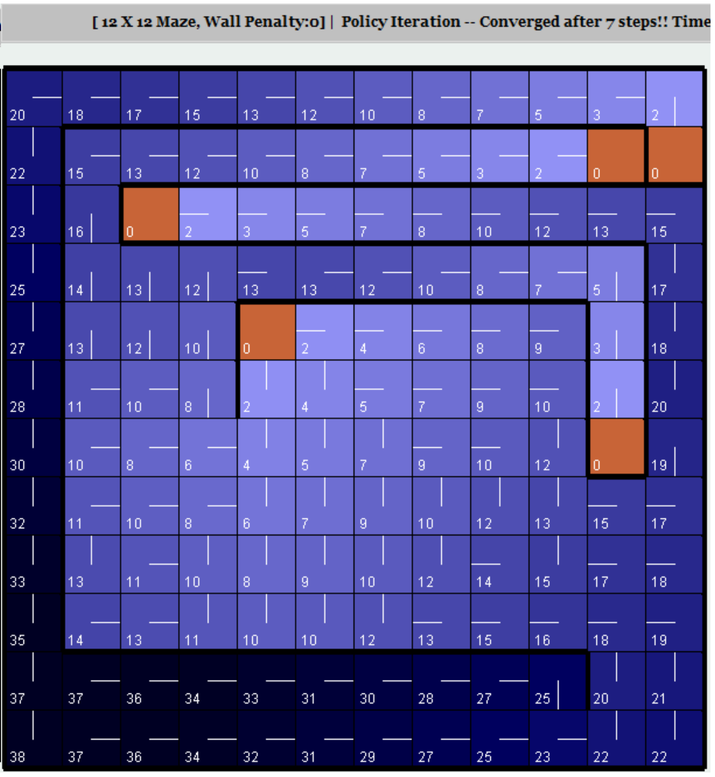
220	202	183	165	147	128	110	92	73	55	37	18
238	83	101	116	108	91	73	55	37	18	0	0
256	65	0	18	37	55	73	92	110	128	142	134
272	47	38	44	62	80	97	106	91	73	55	118
275	37	27	34	0	10	19	25	31	45	37	99
260	31	22	25	9	11	14	17	21	31	18	81
242	28	18	15	12	13	15	17	20	28	0	63
224	28	19	16	14	15	17	18	21	24	34	44
206	33	23	19	17	18	19	21	23	26	30	40
187	46	33	28	26	26	28	30	32	35	35	44
169	157	147	138	128	118	109	100	91	87	46	54
172	157	146	135	125	114	104	93	82	70	58	68


[12 X 12 Maze, Wall Penalty:0] | Value Iteration -- Converged after 70 steps!! Time

20	18	17	15	13	12	10	8	7	5	3	2
22	15	13	12	10	8	7	5	3	2	0	0
23	16	0	2	3	5	7	8	10	12	13	15
25	14	13	12	13	13	12	10	8	7	5	17
27	13	12	10	0	2	4	6	8	9	3	18
28	11	10	8	2	4	5	7	9	10	2	20
30	10	8	6	4	5	7	9	10	12	0	19
32	11	10	8	6	7	9	10	12	13	15	17
33	13	11	10	8	9	10	12	14	15	17	18
35	14	13	11	10	10	12	13	15	16	18	19
37	37	36	34	33	31	30	28	27	25	20	21
38	37	36	34	32	31	29	27	25	23	22	22


[12 X 12 Maze, Wall Penalty:50] | Policy Iteration -- Converged after 7 steps!! Tim

220	202	183	165	147	128	110	92	73	55	37	18
238	83	101	116	108	91	73	55	37	18	0	0
256	65	0	18	37	55	73	92	110	128	142	134
272	47	38	44	62	80	97	106	91	73	55	118
275	37	27	34	0	10	19	25	31	45	37	99
260	31	22	25	9	11	14	17	21	31	18	81
242	28	18	15	12	13	15	17	20	28	0	63
224	28	19	16	14	15	17	18	21	24	34	44
206	33	23	19	17	18	19	21	23	26	30	40
187	46	33	28	26	26	28	30	32	35	35	44
169	157	147	138	128	118	109	100	91	87	46	54
172	157	146	135	125	114	104	93	82	70	58	68

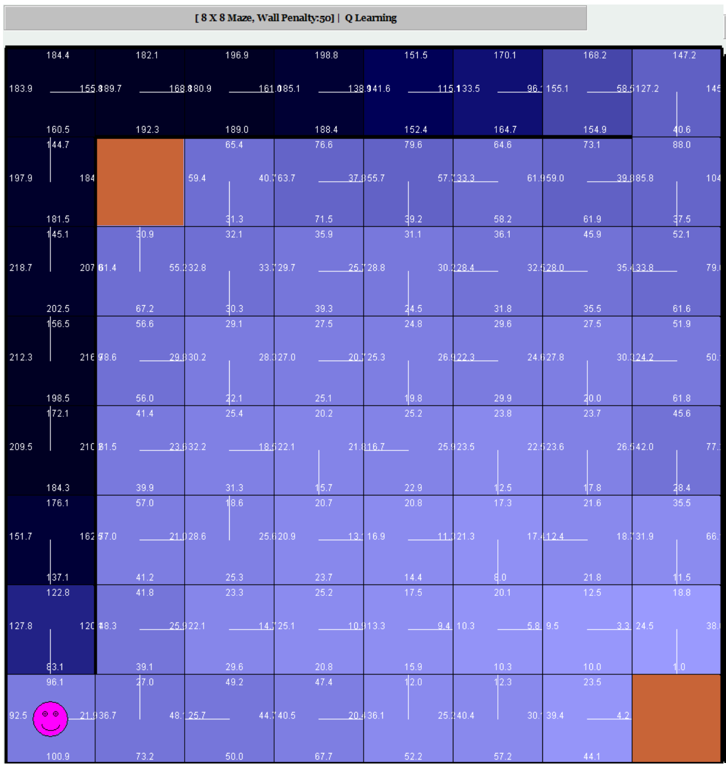

[12 X 12 Maze, Wall Penalty:0] | Policy Iteration -- Converged after 7 steps!! Time

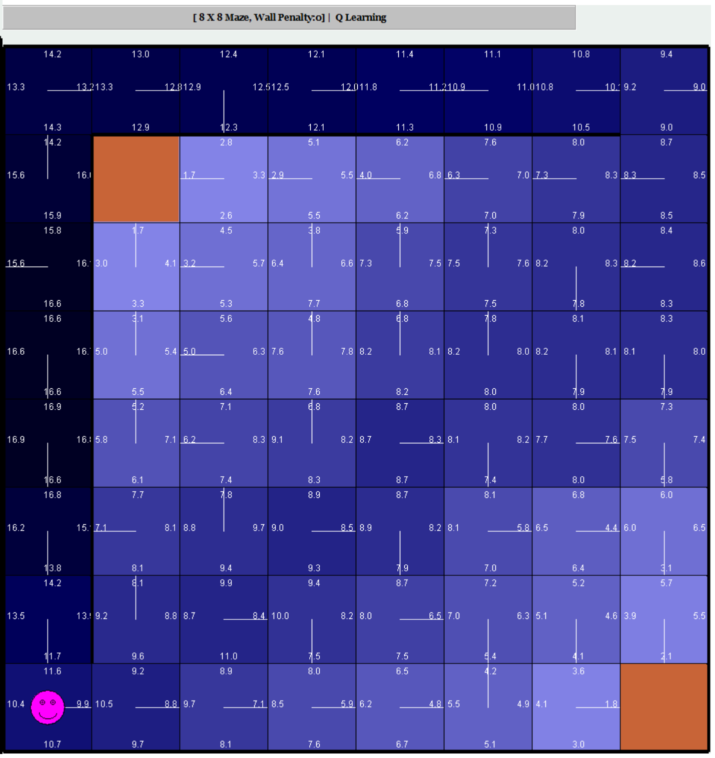
20	18	17	15	13	12	10	8	7	5	3	2
22	15	13	12	10	8	7	5	3	2	0	0
23	16	0	2	3	5	7	8	10	12	13	15
25	14	13	12	13	13	12	10	8	7	5	17
27	13	12	10	0	2	4	6	8	9	3	18
28	11	10	8	2	4	5	7	9	10	2	20
30	10	8	6	4	5	7	9	10	12	0	19
32	11	10	8	6	7	9	10	12	13	15	17
33	13	11	10	8	9	10	12	14	15	17	18
35	14	13	11	10	10	12	13	15	16	18	19
37	37	36	34	33	31	30	28	27	25	20	21
38	37	36	34	32	31	29	27	25	23	22	22

Yet again we see the trend that policy iteration and value iteration came up with the same values again, as well as the same paths, and the bigger difference was based on penalties and no penalties. The no penalties one preferred the center opened goal state as well as two of the goal states not two far into the corridor, while the penalized one preferred only the one in the middle. Based on the two maps thus far, policy iteration performs just as well in terms of results, or rather, produces the same results, as value iteration for both maps, but does so in less than 10% of the time. It may be because they are designed for more complicated problems and despite the increase in states, walls, goal states, and overall complexity, a grid world with 4 actions and only one type of penalization is not enough to distinguish the two algorithms from each other, as at their core they both use Bellman's equation and they differ the most in terms of how policy is decided and when the actual algorithm converges, since they are aware of the unities and rewards, making them similar in this environment.

Q- Learning

For the third algorithm, I chose Q- learning. This algorithm uses model- free approach, where instead of learning about the different values about the state, it will make decisions on immediate actions and how their outcomes, learning through trial and error.

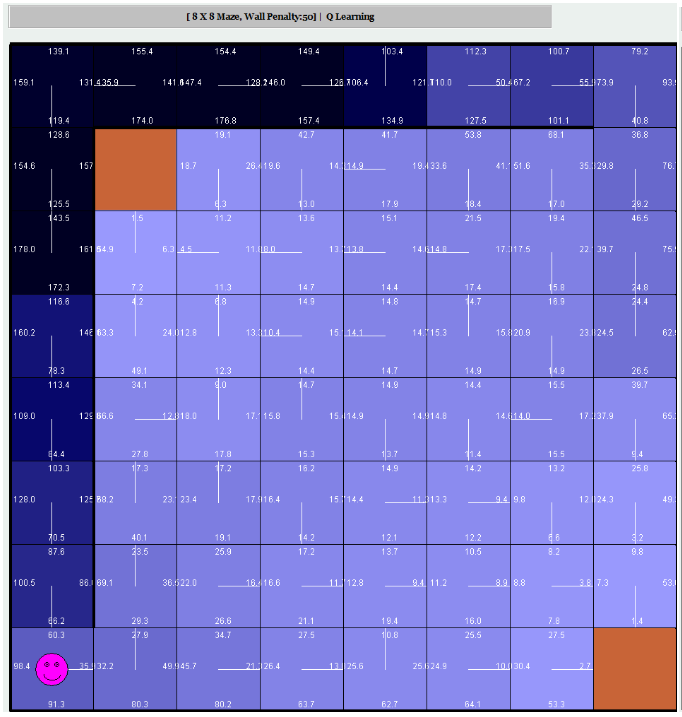


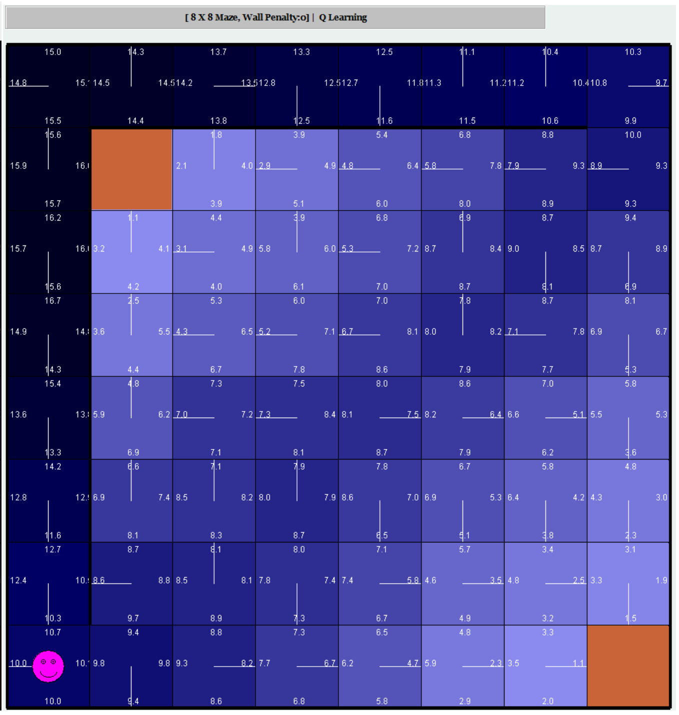


This is ran with a decaying learning rate, so that the longer the program runs, the less impact new updated value has on the overall value. We also ran it with a PJOG of .3 for default, as well as an epsilon of .1. The epsilon provided some form of restart for us to pick a different action just in case the algorithm gets trapped in a local optima. We ran the algorithm for 1000 iterations, since the map is relatively smaller and more straight forward, it was able to converge in that time. The penalized map did not take the most optimal route, but was still able to arrive at the goal within a reasonable amount of time, but the no penalized map has a few cases where if you followed the policy to a certain spot, the current and next state will have conflicting policies that creates a loop, going between the two locations indefinitely unless you randomly make a wrong move.

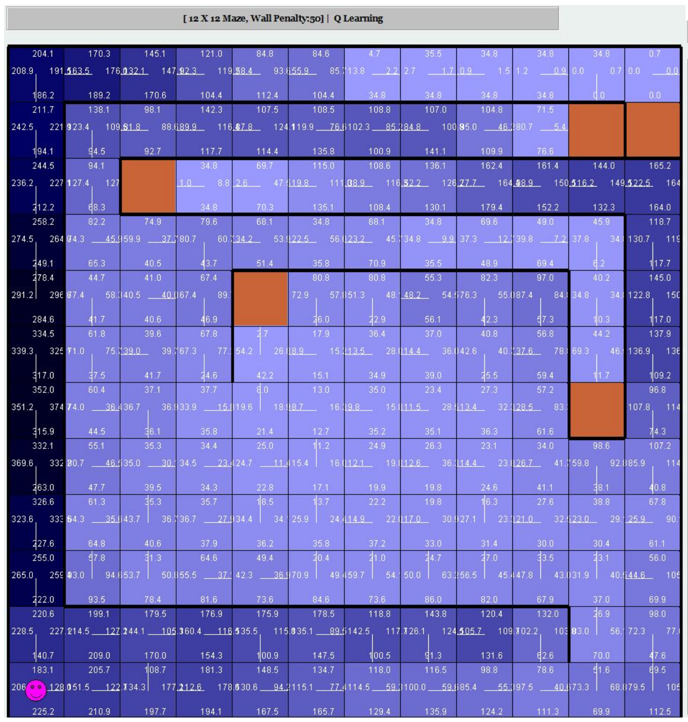
This loop shows how much a model- free learner suffers compared to that of policy and value iteration in this environment, where more iterations are needed for more accurate convergences, taking over 1000 where the other two algorithms completed the task in 1% and .1% of the iteration.

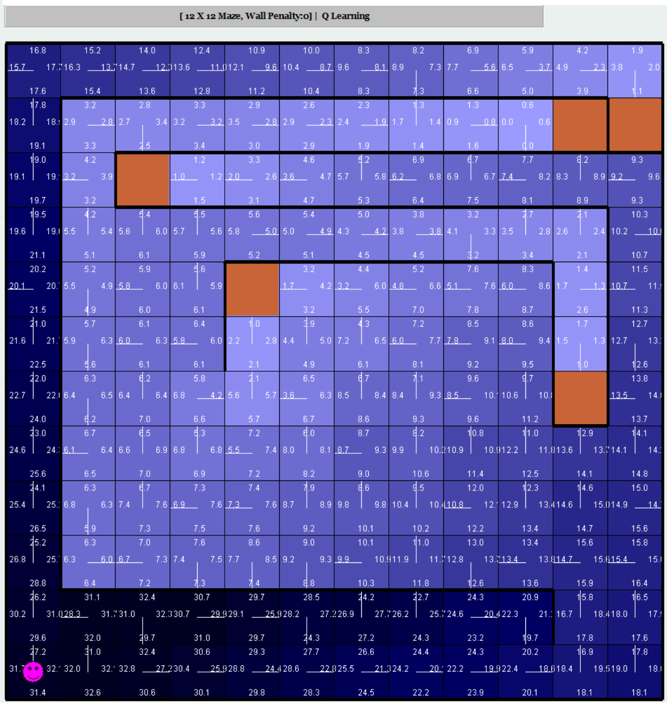
The gradient is also an indication of how much the area was explored, which the corner hallway is very dark due to the lack of options to leave the area once entered, but with no penalties, we can see the middle diagonal with no walls are also explored heavily, presumably because the lack of penalty to push the agent towards the goal, adding to the iterations needed but increasing exploration.



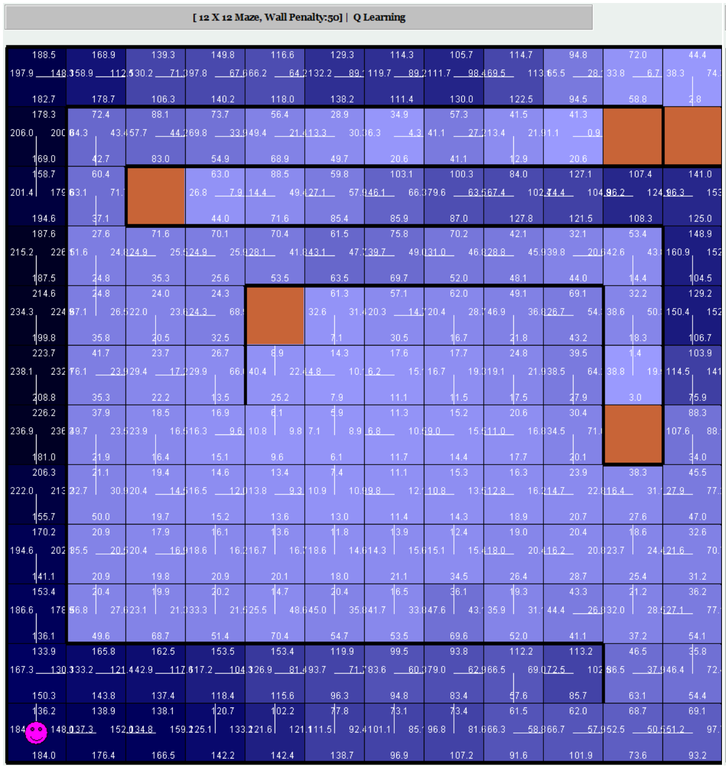


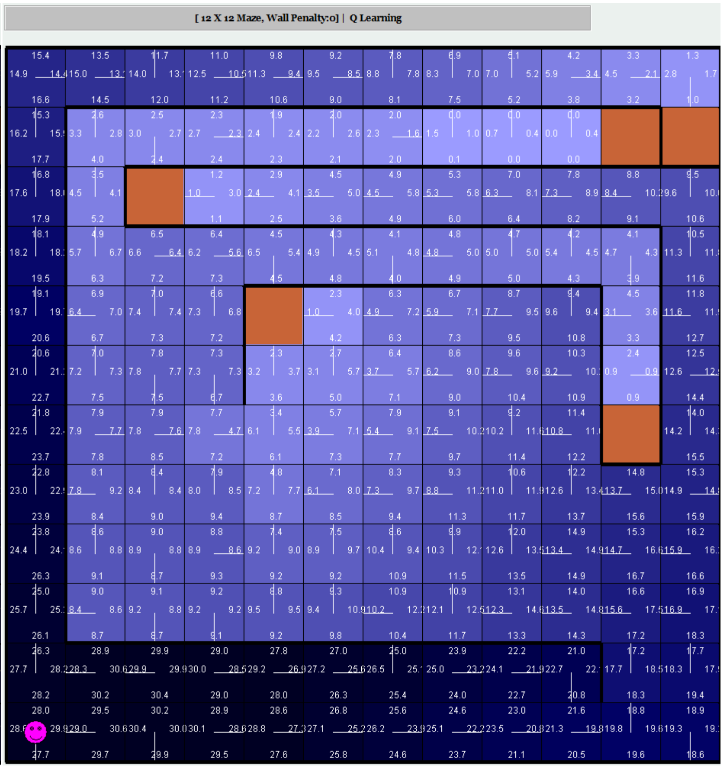
Here we tested the same parameters except increasing epsilon to .4. We can see that the gradient has become more spread out for the no penalized one, while areas in general became less explored with less dark gradients in penalized. It can be hypothesized that having less emphasis on exploration have similar affects as having penalization, pushing the agent towards the correct answer greedily, while no penalization and more exploration resulted in more area having heavier gradients, putting less focus on the correct path and making many poorer policies, such as running into the edge or walls.





With a larger map, it becomes clear that Q- learning is unable to produce the most optimal path, at least with a low number of iterations. With epsilon at .1, the penalized map was barely able to find a sub optimal path with the walls pushing it, while the none penalized one made a lot of dead ends that ran into walls or ran in loops with its neighbors. It may be attributed to the lack of iterations that didn't allow the agent to explore some areas thoroughly enough, a fast decision is made where instead of running into a wall the agent will simply run in the opposite direction for immediate rewards.





As we increased the epsilon, the gradient spread out more, and once again, having penalization pushed the agent towards the goal more successfully while no penalization and higher epsilon produced the worst result, with a lot of random exploration and nothing to push the agent towards the goal.

In terms of Q learning, it can be concluded that since we are doing model- free learning, it requires many magnitudes more of iterations in order to produce the same results as policy and value iteration. Another key point is, given this simple grid world environment, even with long corridors that create local maximas, the world does not benefit from random exploration, and a more greedy and straight forward approach works better. Overall Q- learning performed poorly with its less optimal paths and incorrect policies; while these can be improved, it would require even more iterations, and it's the cost of model free learning.

Conclusion

Even in a simple grid world environment, it appears that a model provided creates a huge difference, hence Q learning performing noticeably worse and slower than the other two. Policy and value iteration generally produced similar results, most likely because of the lack of actions available in a grid world, but its run time is significantly shorter in terms of iterations to simply converge on one of the four options. When running with PJOG, with only four options at higher random values, it becomes better just to completely randomize the outcome than to have the agent chose a run policy majority of the time, but with straight forward environments, no incorrect policies performs the fastest.

Penalties and loss of points incentivizes the agent greatly, and with the simple environment, it can push the agent towards the goal state directly due to lack of other smaller rewards or more negative rewards per block. While this modeled real world poorly, it does highlight the importance of penalties and points, and how quickly it is to determine a policy, rather than state value.

Acknowledgements

Deep Reinforcement Learning Demysitifed (Episode 2) - Policy Iteration, Value Iteration and...

Alzantot & Alzantot

https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa

Reinforcement Learning - Simulator

Rohit Kelkar and Vivek Mehta

https://www.cs.cmu.edu/~awm/rlsim/