CS4641: Reinforcement Learning

Paras Jain

Abstract

This report details an exploration of Reinforcement Learning through *Value Iteration*, *Policy Iteration* and *Q-Learning*. For interest of comparision, I also ran a short experiment using *Prioritized Sweeping*. This paper compares Value Iteration and Policy Iteration to determine which situations each is best applicable. Tradeoffs between Value Iteration and Policy Iteration are explored. Analysis will conclude with a look at Q-learning and finally a brief comparision to Prioritized Sweeping.

1 Markov Decision Processes Chosen

Two Markov decision processes were chosen of the same family of Gridworld MDPs (as covered in class). Two problems of the same class were chosen from the same class to compare how problem size affects.

The GridWorld MDP is where an agent is on a grid of squares and the goal is to reach some goal state. In order to encourage the agent to converge to a solution, there is a penalty of -1 for being in a state. Hitting a wall has a penalty of -50.

1.1 Problem instances

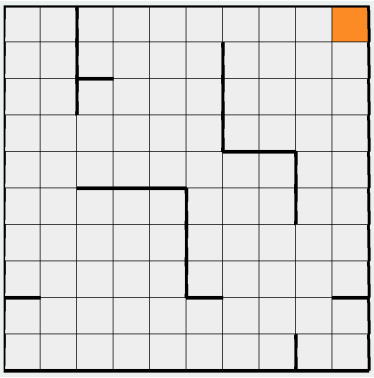

Figure 1: Medium map. Goal state in orange.

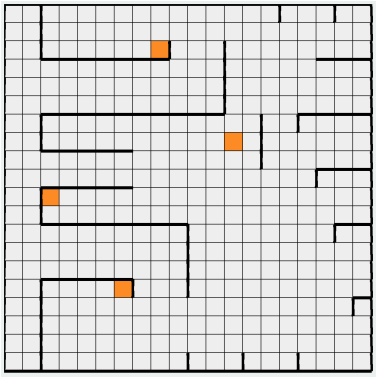

Figure 2: Large map. Goal state in orange.

Each state has a transition function Equation 1, parametrized by probability p which represents the probability of the transition not going as intended.

$$S_{n+1}(x) = \begin{cases} x, & \text{with probability } 1-p \\ rotate(x, 90°), & \text{with probability } \frac{p}{3} \\ rotate(x, -90°), & \text{with probability } \frac{p}{3} \\ rotate(x, 180°), & \text{with probability } \frac{p}{3} \end{cases} \tag{1}$$

The medium map was designed to be relatively small with wide spaces with limited numbers of walls - refer to Figure 1. It is 10×10 with 100 total states. There is a single goal.

The large map was designed to have avenues with walls of varying configurations (varying length/width/direction avenues) - refer to Figure 2. It is 20×20 with 400 total states.

1.2 MDP Interest

I chose Gridworld as it is the most common example of reinforcement learning that students learn. The problem has received a huge amount of research attention and are relatively well understood. By choosing two instances of the same MDP class, we can compare the effect that board size has on the results.

For the large map, there are multiple goals in comparison to a single goal with the small map, which will be interesting to compare to the medium map. The placement of the goals is generally near walls which makes the direction of approach to a goal important. These walls make solving for an optimal policy much more difficult as there are channels through which the actor has to move through. These channels also will test exploration versus exploitation as if the agent does not explore enough, it will not navigate down the channel towards the goal.[1]

The multiple avenues should reveal some interesting conclusions regarding exploration versus exploitation. Note that there are various traps around some of the goals where some goal states are next to walls and corners which would make approaching the goals riskier than the goal in the open.

2 Phase 1 - Value Iteration/Policy Iteration

2.1 Value Iteration

Refer to Table 1, Figure 5 and Figure 6 for results.

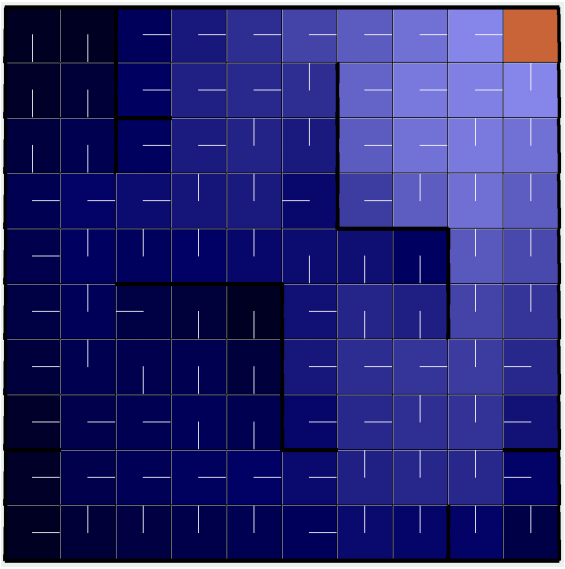

Figure 3: Value Iteration policy for medium map with $p = 0.2$

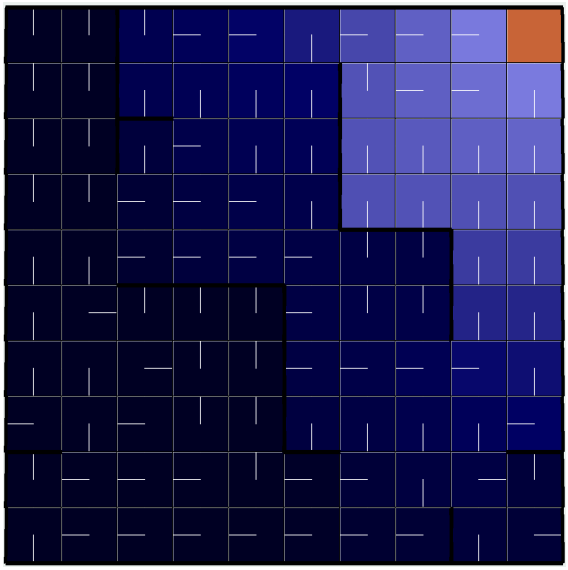

Figure 4: Value Iteration policy for medium map with $p = 0.8$

[1] See the Hallway Problem

Medium map		
p	**Iterations**	**Time taken (ms)**
0.0	19	51
0.1	40	20
0.2	62	29
0.3	90	48
0.4	132	85
0.5	203	89
0.6	417	196
0.7	1821	907
0.8	2178	1216
0.9	679	299
1.0	708	332

Large map		
p	**Iterations**	**Time taken (ms)**
0.0	18	60
0.1	37	128
0.2	54	173
0.3	80	259
0.4	127	411
0.5	193	622
0.6	420	1397
0.7	1503	5113
0.8	1760	5910
0.9	678	2224
1.0	471	1560

Table 1: Value iteration results for both maps. Note that p is the probability of state transition error.

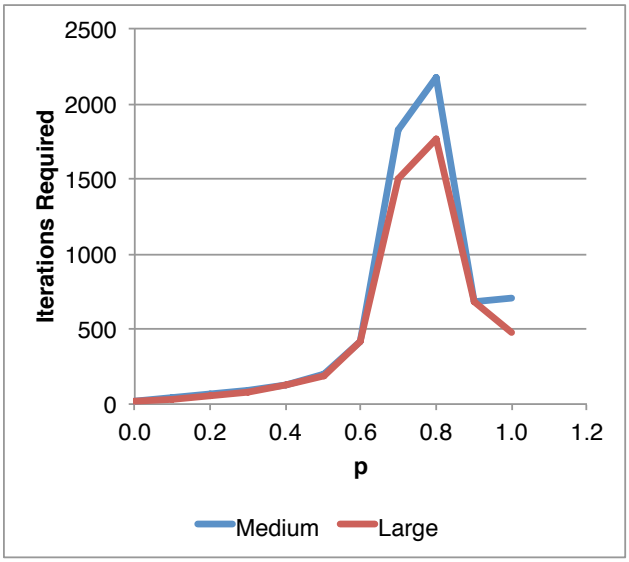

Figure 5: Iterations to convergence for Value Iteration for both maps.

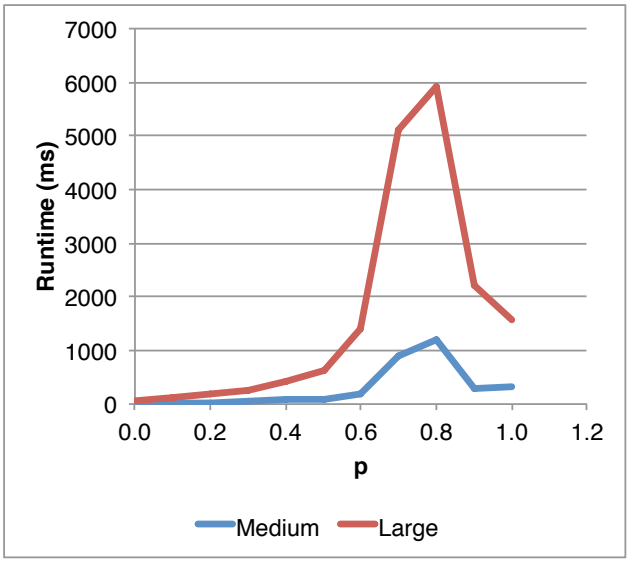

Figure 6: Runtime to convergence for Value Iteration for both maps.

Value iteration performed well on both gridworld problems. For value iteration, both runtime and steps taken were highly correlated to the value of p in the MDP. Interestingly, the number of iterations to convergence was similar between the two map sizes. However, it appeared that the large map took significantly longer to converge than the medium map - this likely is just due to increased time taken per iteration which makes sense given that there are simply more states to process.

I suspect that the peak at $p = 0.8$ is because the transition model is essentially uniformly random across each of the 4 possible states. Optimizing this random behavior is understandably difficult, especially given how value iteration works.

Compare Figure 3 and Figure 4 to see how generated policies varied between $p = 0.2$ and $p = 0.8$. For $p = 0.8$, the policy seems to avoid walls and then seems to generally choose the action

that brings the agent closer to the goal. For $p = 0.8$, it seems that the agent chooses to collide with walls and otherwise tries to escape from the goal. This seems to not make any sense but consider that the agent has a lower probability of successfully transitioning to the desired state than transitioning to an undesired state. As it is more confident that it will fail, value iteration chose to do the opposite action.

2.2 Policy Iteration

Refer to Table 2, Figure 9 and Figure 10 for results.

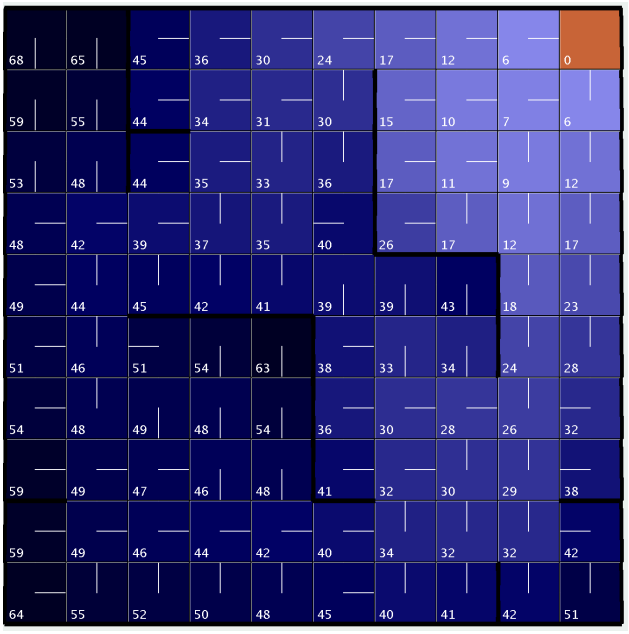

Figure 7: Policy Iteration policy for medium map with $p = 0.2$

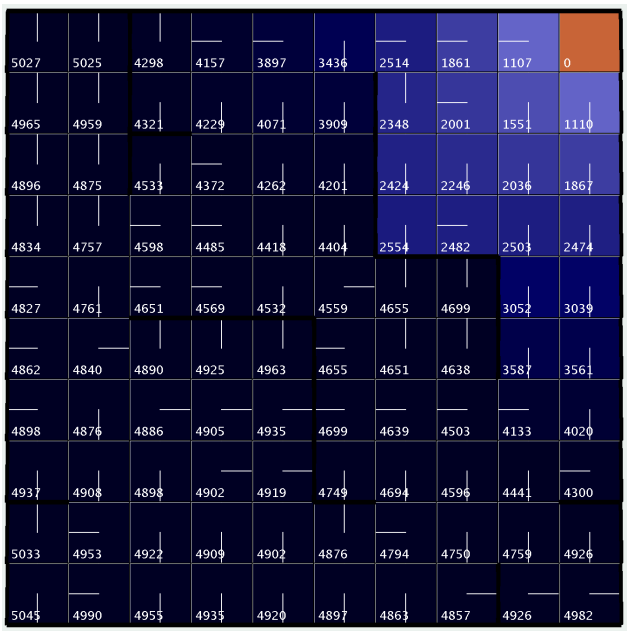

Figure 8: Policy Iteration policy for medium map with $p = 0.8$

Medium map		
p	**Iterations**	**Time taken (ms)**
0.0	11	84
0.1	6	155
0.2	7	105
0.3	7	59
0.4	6	75
0.5	15	82
0.6	5	129
0.7	5	164
0.8	23	49
0.9	24	199
1.0	16	213

Large map		
p	**Iterations**	**Time taken (ms)**
0.0	15	692
0.1	13	672
0.2	18	451
0.3	15	523
0.4	9	340
0.5	8	560
0.6	6	961
0.7	7	1310
0.8	57	1655
0.9	312	2888
1.0	18	1625

Table 2: Policy iteration results for both maps. Note that p is the probability of state transition error.

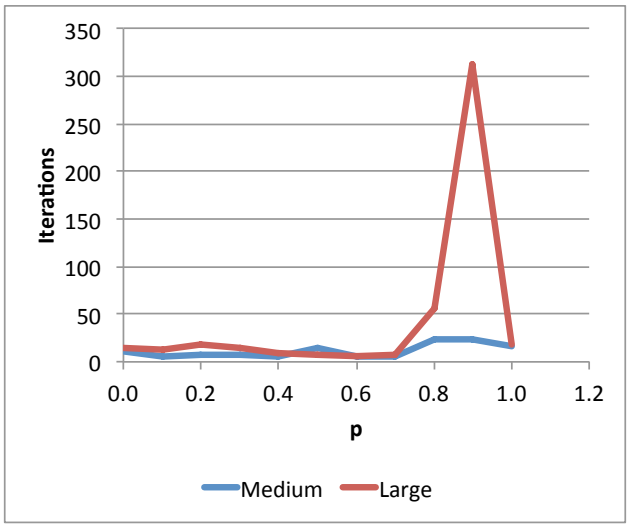

Figure 9: Iterations to convergence for Policy Iteration for both maps.

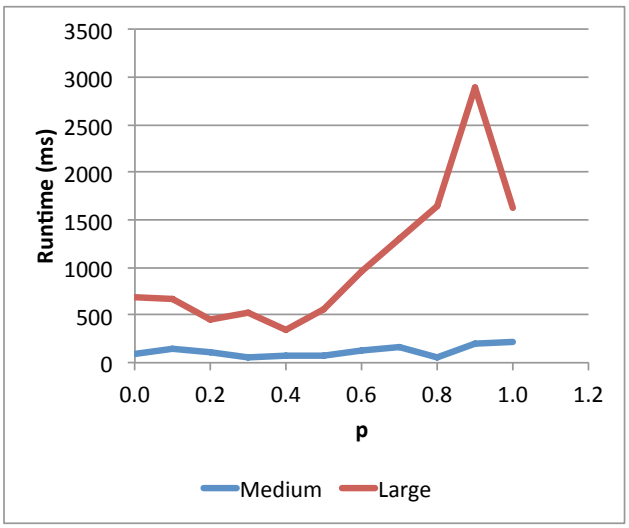

Figure 10: Runtime to convergence for Policy Iteration for both maps.

Algorithm	**% Increase in Runtime**
Value Iteration	450%
Policy Iteration	840%

Table 3: Asymptotic behavior of algorithms explored by examining the percentage increase in runtime from the medium map to the large map.

Policy iteration seems to be much more stable across the choice of p with exception with one outlier. The total number of iterations to convergence is much lower for policy iteration. Runtime also is similarly relatively stable with the exception of one outlier at $p = 0.7$ for the large map. Iterations seemed to increase for values of p close to 1. This makese sense as this would mean the agent almost never does the intended action. It is interesting to see that runtime was slightly higher for $p = 0$ for the large map which meant that the agent always completed the intended action. Perhaps this is due to limited exploration in a exploration vs exploitation tradeoff where it rarely completed sub-optimal actions.

Compare Figure 7 and Figure 8 to see how generated policies varied between $p = 0.2$ and $p = 0.8$. For $p = 0.2$, the results are relatively straightforward. $p = 0.8$ generated an inverted policy where it seems to be taking a less probable sub-optimal action with the goal of obtaining a desired state transition. Like value iteration, policy iteration seemed to take advantage of this trick.

2.3 Value iteration versus Policy Iteration

Overall, policy iteration was significantly faster to run to convergence than value iteration for the Gridworld MDPs. It also appears to have fewer pathological cases of p as compared to value iteration. When deciding which problem to use, consider how the problem domain scales - value iteration has better asymptotic runtime as compared to policy iteration (refer to Table 3).

Comparing generated policies (Figure 3 and Figure 4 versus Figure 7 and Figure 8), both algorithms have very similar policies. Both algorithms were able to utilize the fact that attempting

a low probability sub-optimal transition could result in an optimal transition.

Policy iteration appeared to take much fewer iterations to converge as compare to value iteration. Policy iteration does much more work in each iteration than value iteration which explains why it's runtime is not absurdly smaller than value iteration. Policy evaluation is expensive but it seems to be worth it as overall, policy iteration was faster than value iteration.

Comparing the algorithm performance as a function of p, value iteration is strong at situations where action is definite (either high p or low p). Value iteration has pathological performance in situations where the transition state is very ambiguous. This makes Policy Iteration much better suited to situations where there is a non-negligible amount of noise (in say sensor readings or localization). I would suspect therefore that Policy Iteration would work much better for real-world RL robots. However, value iteration did produce more optimal solutions for extremely ambiguous transition models (about $p0.7$). This runtime-performance tradeoff is something to consider when choosing an algorithm with extreme amounts of noise.

This analysis aligns with the intuition that policy iteration is better for situations where the transition model is large with many possible actions. However, value iteration excels where the state space is large or grows non-linearly.[2]

3 Phase 2 - Q-Learning and Prioritized Sweeping

3.1 Q-Learning

3.1.1 Parameter Hyperspace Analysis Experiment

To study Q-learning, I modeled the parameter space to analyze the how parameters affected Q-learning. I explored a hyperspace of parameters, runtime and score. Out of some initial tests, I determined to focus analysis on a subset of parameters and the effect on two key performance metrics of a RL algorithm - runtime and utility achieved.

Iterations was fixed at 10,000 in this case in order to be able to visualize the highly-dimensional parameter space to produce meaningful conclusions. Some experiments were run over 20 trials with the results averaged in order to reduce the effect of randomization.

3.1.2 Parameter Hyperspace Analysis Experiment Results

Refer to Figure 13[3] and Figure 16[4] for visualization of the parameter space over p, ϵ and scores. Score in these charts is the value function where smaller values are better.

For the medium map (Figure 13), there seems to be a linear relationship between the value function, p and ϵ. This effect is clearer on the heatmap (Figure 14). For the large map (Figure 16), the relationship between p seems to be very weakly linear with smaller p values leading to slightly better policies. There appeared to be little relationship between ϵ and score.

Smaller p values produced significantly better policies for both maps.[5] The relationship with p was expected as that simply means less noise and more certainty in the transition model. Q-learning is especially sensitive to this given that is is a model-free learner.

Smaller ϵ produced slightly better policies in the medium map. This appears to indicate that for the smaller map, exploitation was suffient to produce good policies and that exploration was

[2]Consider gridworld where the states space grows at $O(n^2)$ where n is the length of the grid

[3]For an interactive 3D scatter plot, see `https://plot.ly/~parasj/4.embed`

[4]For an interactive 3D scatter plot, see `https://plot.ly/~parasj/6.embed`

[5]This may be hard to see on the scatterplot for the big map. Refer to the online interactive version to see this more clearly.

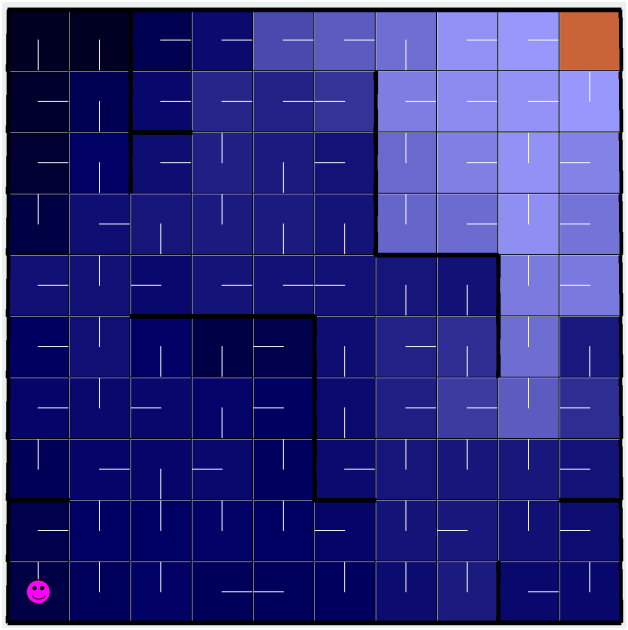

Figure 11: Q-learning policy after 10,000 iterations with $p = 0.3$, $\epsilon = 0.1$

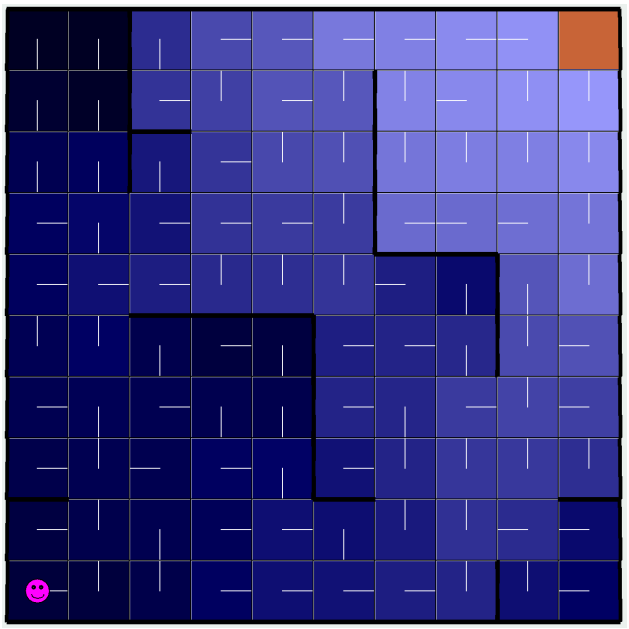

Figure 12: Q-learning policy after 10,000 iterations with $p = 0.7$, $\epsilon = 0.1$

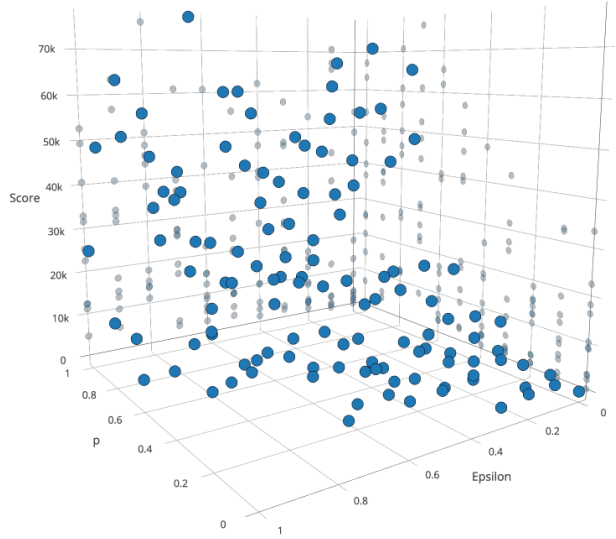

Figure 13: Q-Learning scores for the medium map over a space of p and ϵ values, visualized as a 3D scatter plot.

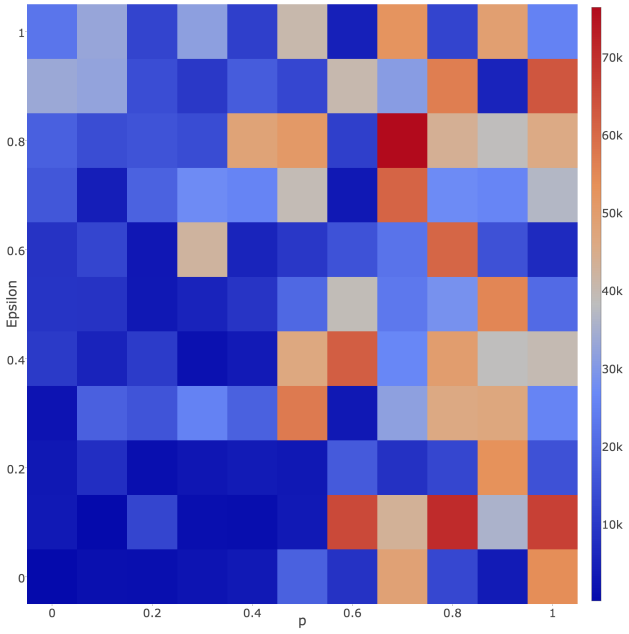

Figure 14: Q-Learning scores for the medium map over a space of p and ϵ values, visualized as a heatmap.

not neccessary. This makes sense given that there are few traps in the medium map. For the large map, values around 0.8 produces the best policies. For a large map with traps and multiple goals, exploration was advantageous as compared to strict exploitation. However, too much exploration produced worse results.

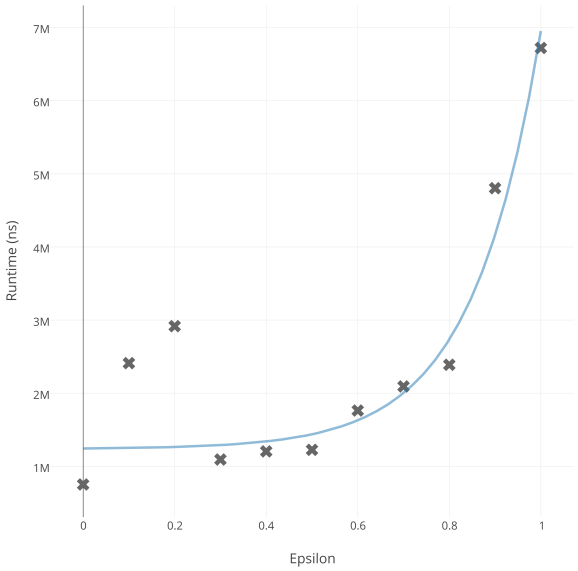

Figure 15: ϵ vs runtime for the medium map for Q-learning. An exponential regression is superimposed on the graph.

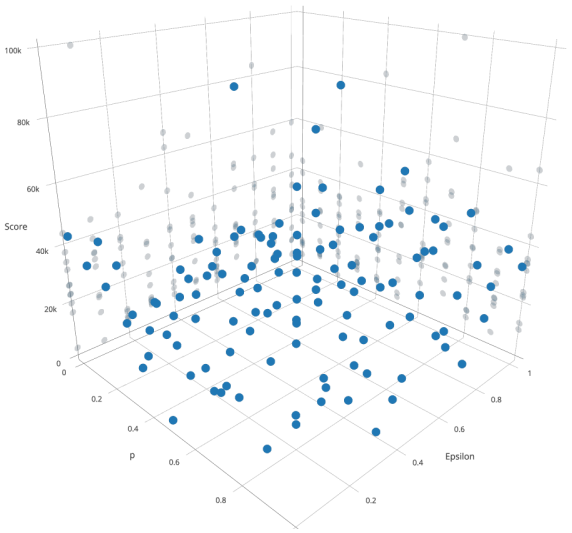

Figure 16: Q-Learning scores for the big map over a space of p and ϵ values, visualized as a 3D scatter plot.

ϵ had a large impact on runtime - see Figure 15. There was an exponetial relationship between ϵ and runtime. This makes sense as exploration is about trading some controlled amount of runtime performance for some amount of exploration (ϵ-greedy follows a Boltzmann distribution). However, this tradeoff is important to consider if the RL algorithm is running in a runtime-constrained environment.

3.1.3 Comparision of Q-learning with model-based learners

Generated policies
Q-learning on the medium map produced lower quality results as compared to model-based learners. This is expected as the model-based learners have information regarding the problem rewards and probabilities of the transition model. Q-learning at times produces strange results for certain states where many more iterations are required to converge.

Runtime
Q-learning takes significantly longer to converge when compared to model-based learners, both in terms of iterations as well as time. This makes sense as Q-learning has no starting knowledge regarding the problem. This demonstrates that Q-learning is trading learning time for a priori knowledge. These problems also had deterministic rewards - for situations with stocastic rewards, runtime could be pathologically worse as Q-learning would require repeated visits to a state.

Effect of p
Q-learning seems to be significantly more sensitive to the value of p as compared to model-based learners. This makes sense as the Q-learner is effectively trying to learn the transition model probabiities through increased training time. Given a fixed number of iterations, it makes sense that Q-learning performance was heavily dependent on the choice of p.

Exploration versus Exploitation
For simpler maps with few traps, exploration was not that advantageous. For complex maps with many traps and multiple goals, increased exploration (to a point) produced better maps. Given a set MDP problem, the main lever for controlling Q-learning performance is the ϵ value. Deciding the amount of exploration is the key choice in a reinforcement learning problem. Choosing ϵ depends on how many states are in the MDP and how complex rewards are (more complex MDPs can benefit from increased exploration).

3.2 Prioritized Sweeping

Prioritized Sweeping is another model-based algorithm that has very interesting temporal properties as a result of its use of Bellman Backups. This paper explores the hyperspace of parameters, score and runtime. Figure 17[6] shows the relationship between score, p and ϵ. Figure 18[7] explores the relationship between p, ϵ and runtime.

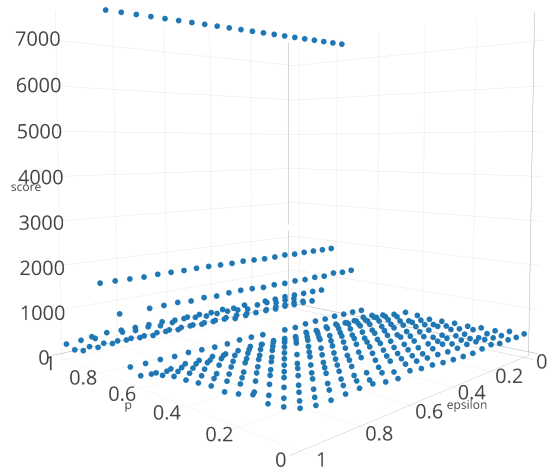

Figure 17: Relationship between p, ϵ and score for the medium map with Prioritized Sweeping.

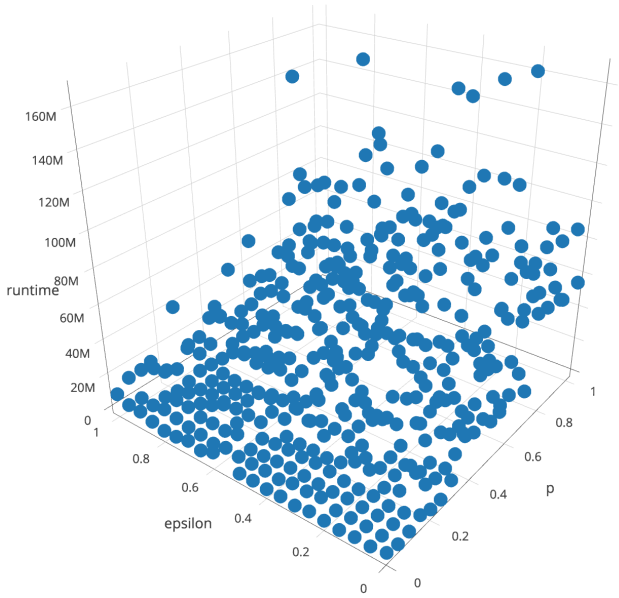

Figure 18: Relationship between p, ϵ and runtime for the medium map with Prioritized Sweeping.

3.2.1 Effect of p and ϵ on score

Figure 17 is interesting as it seems ϵ has little impact on the score. However, it seems Prioritized Sweeping is very sensitive to the value of p. Moreover, it seems Priortized Sweepign has pathological cases near $p = 0.7$ which is similar to Value Iterations pathological cases. I suspect this is for the same reason as Value Iteration where these cases are where the transition model is near random and therefore there is little that the algorithm can do to optimize a policy.

[6]For an interactive 3D scatter plot, see `https://plot.ly/~parasj/12.embed`
[7]For an interactive 3D scatter plot, see `https://plot.ly/~parasj/13.embed`

3.2.2 Effect of p and ϵ on runtime

Figure 18 is interesting when compared with the results from Q-learning. ϵ appears to have little impact on runtime which contrasts with Q-learning where it was ϵ that had an exponential correlation with runtime. However, unlike other model-based RL algorithms like Value Iteration and Policy Iteration, p appears to exponentially correlate with runtime. As Prioritized Sweeping is a model-based learner, this was unexpected but makes sense as more uncertainty would lead to a tree-like effect when exploring the state space where exponentially more states need to be explored.

4 Conclusion

4.1 Value Iteration

Value iteration is effective overall and is reasonably quick at solving the optimal policy. It used an unexpected trick to optimize for situations where the probability of a succesful transition was low. However, value iteration took many iterations to converge and also had some pathological cases where the transition model was ambiguous. As value iteration is a model-based learner, it is only well suited to scenarios where the transition probability is known. Experiments showed that value iteration had good asymptotic performance for problems where the state space grew non-linearly.

4.2 Policy Iteration

Policy iteration worked reasonably well for the Gridworld MDPs. Policy iteration used the same trick as value iteration and generally produced similar optimal policies to value iteration. It has lower constant-factor runtime constants so for smaller problems, it was faster than value iteration. However, its asymptotic runtime growth was worse than value iteration. Policy iteration did not seem to have as many pathological cases as value iteration which makes it well suited for ambiguous transition models. As it is a model-based learning algorithm like value iteration, it requires a priori knowledge of the state transition probabilities.

4.3 Q-learning

Q-learning trades some runtime performance for a priori information as it is a model-free learner and does not require the probability of a state transition. However, Q-learning takes many more iterations to converge as compared to a model-based algorithm and requires longer runtime. For Q-learning, results are very sensitive to the choice of ϵ which controls the exploitation-exploration tradeoff. Larger ϵ values cost exponentially more runtime but can produce more optimal policies for more complex MDPs. While Q-learning is less effective than value iteration or policy iteration, it does not require a priori knowledge of p.

4.4 Prioritized Sweeping

Prioritized sweeping was explored briefly to compare to the other model-based algorithms. It has performance characteristics similar to value iteration (which makes sense as it is essentially an incremental form of value iteration). Runtime is heavily dependent on p but for MDPs which have low noise, prioritized sweeping can be an efficient way to solve for the optimal policy.