Markov Decision Processes

In this paper, I will be analyzing the strengths and weaknesses of Value Iteration (VI) and Policy Iteration (PI) based on couple Markov Decision Processes (MDP). Finally, I will apply Q-Learning (QL), a type of Reinforcement Learning algorithm, to solve the same problems. All problems are done using the BURLAP library provided by James MacGlashan of Brown University.

MDP Problems Chosen

MDP is a type of problem in which an agent interacts with the environment containing a set of possible states. At each time step, the agent performs an action to move to a different state from its current state, which changes the environment state and the agent possibly receives a reward/penalty from the environment. The goal of the agent is to discover an optimal policy, defined as the set of actions to do in each state, such that it maximizes the total value of rewards received from the environment in response to its actions. Our goal is to solve the MDP problem using the planning algorithms VI and PI. Therefore, all MDP problems are defined by a set of states, set of actions, state transition model which describes the change in the environment based on chosen action current state, reward model which describes the reward received by the agent from the environment after performing an action.

I have chosen to analyze 3 variations of the Grid World problem. The Grid World problem is a 2-dimensional problem of a limited size, where the agent takes action in the form of moving east, west, north and south. The agent can traverse through the entire grid except walls that block the agent. Each of these grids contain a terminating condition, which is the goal state of the problem. Although simple, Grid World problems reveal the strength and weaknesses of RL algorithms based on the size of the grid chosen. To make the problem more interesting, all chosen problems are stochastic; that is, when an agent performs an action, it has an 80% chance to ending up in the intended direction. I will be analyzing 3 domains in this paper:

a. Simple Grid World: This is a 5x5 grid as shown in Figure 1a. The agent, represented by the gray ball, starts at the bottom left corner. Its goal is to reach Georgia Tech's mascot Buzz, who is located at the top right corner. The total number of states are 25.
b. Four Rooms: This is a problem available on Burlap and is as shown in Figure 1b. This is different from the simple grid world in that the 11x11 grid is divided into 4 equal sections. Each room is a 5x5 block in which there is only one non-blocking state, so the agent must get exit through these states to reach the goal state at the top right corner represented by Buzz. This adds a level of complexity to the previous problem as the agent must traverse through specific exits to reach the goal state. There is a high possibility that the agent may get stuck in any of these non-goal rooms. The total number of states in this problem are 121.
c. Maze: Unlike the other two problems, the maze is the most complicated one. It represents a very complicated maze of size 40x40 as shown in Figure 1c. In this case, the agent starts at the bottom right part of the maze, represented by the gray ball and it's goal is the reach Buzz located in the mid-left section of the maze. In addition to the very large number of states, what makes this problem an interesting one are all the blocking walls. This means that the agent's path from its initial state to the terminating state is very restricted; that is, there is very high possibility that the agent may be stuck at a non-terminating position forever.

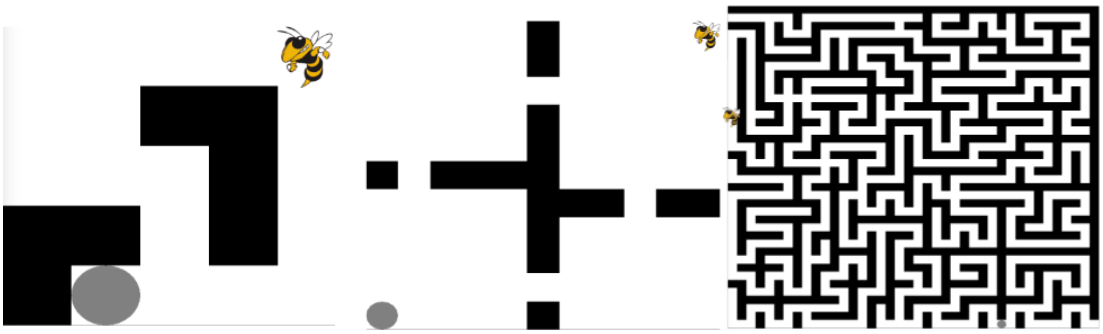

Figure 1 a, b, c: Simple Grid World, Four Rooms and Maze

Grid World problems are simple yet powerful. For the purposes of this paper, they make an excellent choice, as Grid World problems are easy to visualize due to the 2-dimensional space of the problem and relatively limited action space as the agent can only move in 4 directions. However, to keep the problems challenging, I have chosen to analyze the Maze as well. This will help us understand the policy and states of the Grid World domain so we can compare and contrast PI and VI easily. At the same time, its simplicity is powerful in that we can better visualize how tuning the hyper-parameters in QL impact the performance of the agent. Lastly, the smaller run time of Grid World problems make it possible for me to explore a larger space of hyper-parameters so I can really understand how they interact with the problem space.

Parameters Explored

Policy Iteration and Value Iteration: For both of these algorithms, I varied the discount factor (γ), Terminal Reward (TR) and Step Reward (SR). γ is a measure of how impatient the agent is; that is, future rewards are exponentially decayed by a factor of γ. TR is the incentive for the agent to go to the terminating state while SR is the reward at each non-terminating state and is usually negative; therefore, SR is the cost an agent has to pay to move to a state. TR and SR are highly correlated to each other. The higher the value of TR with respect to SR, the most incentive the agent has to move towards the terminating state. On the other hand, a SR would reduce the motivations of the agent to move towards the terminating state as it might benefit more by not having to pay the cost of moving any steps at all. Equation 1 demonstrates the effect of γ on the total reward where T is the length of the episode. These parameters for VI and PI are varied as shown in Table 1.

$$Total\ Discounted\ Reward = \sum_{i=1}^{T} \gamma^{i-1} r_i$$

Equation 1

Parameters	Values
Discount Factor (γ)	{0.99, 0.95, 0.9, 0.8}
Terminating Reward (TR)	{100, 20, 10, 5}
Step Reward (SR)	{-1, -0.1}

Table 1: Parameters Tuned for Value Iteration & Policy Iteration

Q-Learning: Similar to VI & PI, Q-Learning also contains γ, TR and SR. In addition, I varied the learning rate ($\propto$), epsilon (ε) and initial Q-value (Q_{init}). In QL, $\propto$ determines to what extent newly acquired information overrides old information, such that a value of 0 means the agent learns nothing and exclusively relies on prior knowledge while a value of 1 makes the agent consider only the most recent information (ignoring prior knowledge). Q_{init} is the measure of initial condition before the first update occurs; high Q_{init} values are optimistic and encourage exploration. Finally, ε is measure of

randomness in policy. In order to have a good balance between exploration and exploitation, we are going to use an ε-greedy policy. As exploration diminishes over time, the policy used asymptotically becomes greedy and therefore reaches an optimal point. ε is annealed such that it initially encourages a lot of exploration but as training progresses, the agent becomes more confident and chooses actions with maximum Q-values (exploitation). I varied these parameters as shown in Table 2.

Parameter	**Values**
Discount Factor (γ)	{0.99, 0.95, 0.9}
Terminating Reward (TR)	{100, 20, 10}
Step Reward (SR)	{-1, -0.1}
Learning Rate ($\propto$)	{0.5, 0.2, 0.1, 0.01}
Initial Q-value (Q_{init})	{10, 5, 1, 0.1}
Epsilon (ε)	{0.2, 0.1, 0.05, 0.01}

Table 2: Parameters Tuned for Q-Learning

Value Iteration (VI): VI relies on the notion of maximizing the value function of the problem, which represents the utility of a particular state. It is equal to the expected total reward and depends on the policy by which the agent picks actions to perform. Suppose that the agent uses policy π, the value function V^{π} is given Eq 2a. Therefore, the optimal value function V*(s) finds the optimal policy π such that it maximizes V while the optimal policy π* is the policy that corresponds to the optimal value function as shown in Eq 2b, c.

$$V^{\pi} = E[\sum_{i=1}^{T} \gamma^{i-1} r_i]; \quad V^*(\mathrm{s}) = \max_{\pi} V^{\pi}(s) \,; \forall s \in S; \quad \pi^* = \arg\max_{\pi} V^{\pi}(s) \;\; \forall s \in S$$

Equation 2 a, b, c: Value Iteration

Furthermore, we utilize the Bellman equation to obtain the Q-function Q*(s, a), which represents the summation of the immediate reward after performing action a while in state s and the discounted expected future reward after transition to next state s' as shown in Equation 3.

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^*(s')$$

Equation 3: Q-Function

VI computes the optimal sequence of actions by iterative improving the estimate of V(s). VI updates Q(s, a) and V(s) until they converge. It is guaranteed to converge to optimal values.

Policy Iteration: While VI seeks to improve V(s) with each iteration until it converges, PI redefines the policy at each step and computes the value according to the new policy until the policy converges. Since the agent is concerned with finding the optimal policy, the optimal policy sometimes converges before the value function V(s). Both VI and PI are guaranteed to converge to the optimal policy while PI often takes less iterations to converge than VI. Table 3 contains select interesting findings from Simple Grid World, 4 Rooms and Maze. For each problem, I MDP problem and algorithm, I chose the best and worst performing statistics in terms of number of iterations, runtime and steps to exit.

		Simple Grid World (10 states)	Four Rooms (104 states)	Maze (800 states)
Value Iteration	Iterations	16	17	89
	Runtime (ms)	4	18	1244

	Steps to Exit	7	26	154
Value Iteration	Iterations	12	23	164
	Runtime (ms)	73	232	1850
	Steps to Exit	11	30	164
Policy Iteration	Iterations	3	3	12
	Runtime (ms)	6	7	21680
	Steps to Exit	4	6	229
Policy Iteration	Iterations	3	8	9
	Runtime (ms)	83	451	1718
	Steps to Exit	4	23	99999

Table 3: PI & VI on Simple Grid World, Four Rooms and Maze

A clear observation is that the runtime of PI is higher than that of VI. Although the difference is not discernible is a small domain such a Simple Grid World, the difference if almost 12-fold in case of the Maze. Another interesting observation is that PI almost always takes many fewer iterations than VI. For instance, it only takes 12 iterations to converge using MI for Maze while it takes 89 iterations for VI, which is a large difference. However, this gap is mostly visible across larger domain spaces as the difference is not as large in case of Simple Grid World. This implies that PI is certainly more powerful in large problem spaces. Therefore, there is a tradeoff between runtime and number of iterations when it comes to PI. Although I set 100,000 as maximum number of iterations, there are times when both PI and VI fail in case of the Maze. This goes on to show the complexity of the Maze. In Table 4, I am presenting performance of VI and PI on Grid World with varying γ.

		Simple Grid World (TR= 10, SR = -0.1)	Four Rooms (TR = 100, SR = -1)	Maze (TR = 1000, SR = -0.01)
VI (γ = 0.8)	Iterations	9	17	22
	Runtime (ms)	4	28	175
	Steps to Exit	15	27	99999
VI (γ = 0.99)	Iterations	13	28	154
	Runtime (ms)	17	39	1037
	Steps to Exit	9	28	162
PI (γ = 0.8)	Iterations	3	6	6
	Runtime (ms)	6	83	910
	Steps to Exit	4	23	99999
PI (γ = 0.99)	Iterations	3	5	13
	Runtime (ms)	19	878	17300
	Steps to Exit	6	21	223

Table 4: PI & VI on Simple Grid World, Four Rooms and Maze with Varying γ

Right away, one interesting observation in case of Maze is that it doesn't converge with γ = 0.8 for both PI and VI as they perform 99,999 iterations which is the maximum I set for the problem. However, the Maze does converge with a γ = 0.99. By definition, a high value of γ implies that the agent will value future rewards. It is also interesting to note that a higher γ leads to higher runtime for all the problems, albeit small differences steps to exit although the number of iterations is higher. However, I observed that the total reward (relative to TR & SR) gained by the agent is higher with higher γ values in case of the Maze and Four Rooms. However, this is not true in case of the Simple Grid World, where γ = 0.8 is able to yield the highest reward. This implies that in case of Grid World problems, low γ does well on small problem spaces while higher γ does well on

large problem spaces. Based on this observation, I will now explore the impact of tuning TR & SR on PI & VI. The results are as shown in Table 5.

TR, SR	{5, -0.01}	{5, -0.1}	{100, -0.1}	{100, -1}	{20, -0.1}	{100, -0.01}	{500, -0.01}	{10000, -0.01}
	Simple Grid World (VI)				Maze (VI)			
Iterations	9	9	11	11	140	13	151	160
Runtime (ms)	15	18	3	5	1088	14176	1767	1074
Steps to Exit	12	10	9	11	156	211	170	150
	Simple Grid World (PI)				Maze (PI)			
Iterations	3	3	3	3	13	13	12	11
Runtime (ms)	83	101	15	8	13984	14716	21680	24017
Steps to Exit	4	4	3	8	201	211	229	201

Table 5: VI & PI on Simple Grid World Maze with Varying TR, SR

Again, we observe that PI converges before VI in terms of iterations across both domains. One observation that strikes out is the increased runtime of VI on Simple Grid World (SGW) with TR, SR set to {5, -0.01} and {5, -0.1} in comparison to {100, -01} and {100, -1} whereas the opposite is true when the SR is reduced relative to TR in case of Maze. Similarly, both the Maze and Simple Grid World able to exit with fewer steps with TR very large in comparison to SR. However, both Simple Grid World and Maze converge in slightly higher iterations for VI in this case while the reverse is true for the Maze with PI. However, the increases benefit of smaller SR is more noticeable with the Maze. Therefore, we learn that for large grids like the Maze, it's important to set high TR and low SR; otherwise, it may never converge in even 99,999 steps as seen in Tables 3, 4.

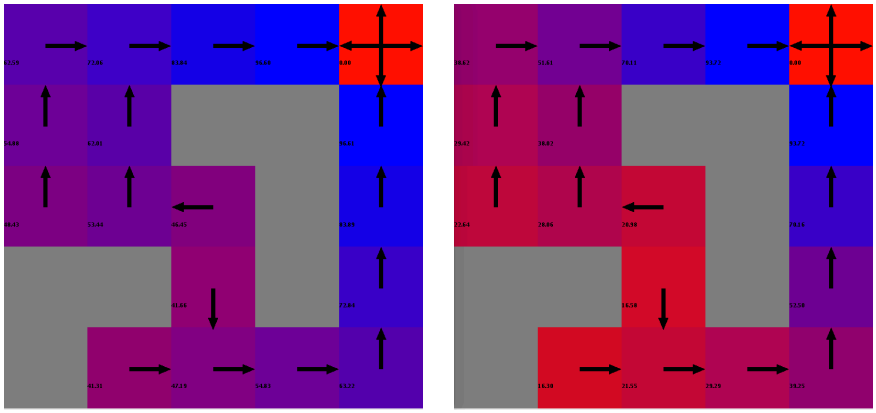

Figure 2 a, b PI on SWG (TR=100, SR=-0.1) with γ=0.9(left) and γ=0.8 (right)

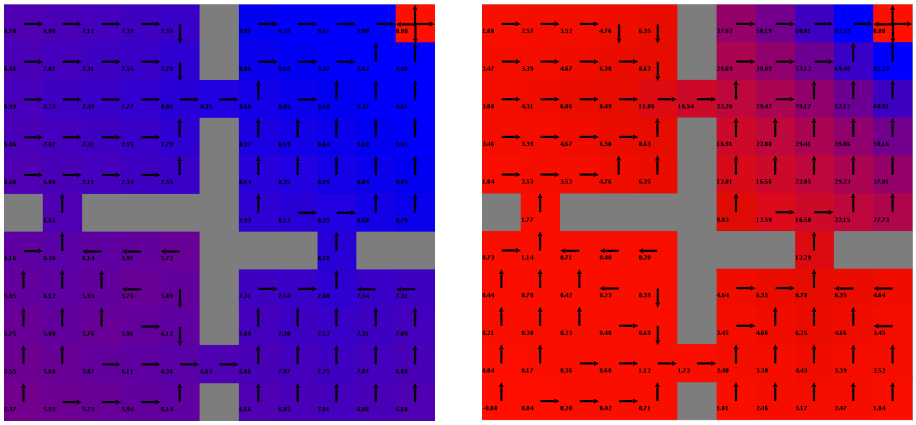

Figure 3 a, b PI on 4 Rooms (TR=100, SR=-0.1) with γ=0.9(left) and γ=0. 8 (right)

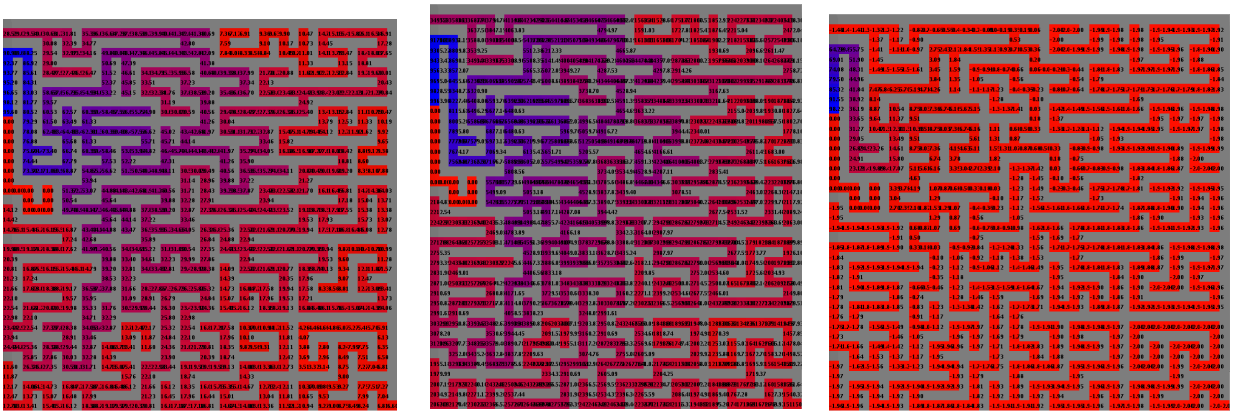

Figure 4 PI on Maze (a) (TR=100, SR=-0.1, γ=0.99) (b) (TR=10000, SR=-0.1, γ=0.99) (c) (TR=100, SR=-0.1, γ=0.95)

Figures 2, 3 and 4 represent how the value changes at each state of the respective grids. Although the values are perhaps not visible, the gradient helps visualize the value of each state with red being relatively low value and blue being high value. In case of Simple Grid World, we notice that choosing $\gamma = 0.8$ leads to a better policy as it distinguishes the optimal path (move right until it hits the wall and then up) from the non-optimal path. However, as the grid size increases, take for instance Four Rooms, $\gamma = 0.8$ makes the path non-distinguishable as all 3 rooms except the goal room have very low values; thus, the agent will likely take longer to reach the goal state. With a higher γ=0.9, it becomes clear however that the agent must move to either of the 2 adjacent rooms from the initial bottom left corner room to reach the goal state. This difference becomes even clearer with a larger grid, the Maze. Consider Figure 4 (a) with $\gamma = 0.99$ and (c) $\gamma = 0.95$. Although the difference in γ is small, it becomes clear that $\gamma = 0.95$ doesn't lead to an optimal policy as the entire maze has very low value function except very close to goal state. However, $\gamma = 0.99$ produces a better policy as it's likely to guide the agent better to the goal state as it distinguishes good states from bad ones as is clear from the color gradient. Finally, I varied TR and SR on Maze as shown in Figure 4 (a), (b). Although the difference is rather minor, it's clear that choosing a higher TR relative SR leads to a better policy.

Q-Learning

Finally, I explored the performance of Q-Learning (QL) on the same Grid problems by varying $\propto$, ε and Q_{init} as discussed earlier. The agent now doesn't know apriori the effects of its actions on the environment i.e. the state transition and reward models are not known. The agent actively learns through experience of its interaction with the environment. In QL, the agent doesn't try to learn explicit models of the environment

state transition and reward functions but discovers the optimal policy by trial and error. An important concept in QL is the exploration v/s exploitation dilemma in which the agent needs to decide whether to trust learnt values of Q(s, a) enough to base its actions upon or try new actions in hopes of getting better rewards. We will be assessing QL based on the average number of steps per episode and average rewards per episode. Figure 5 & 6, we see the impact of varying ε on all 3 Grid World problems using Q-Learning.

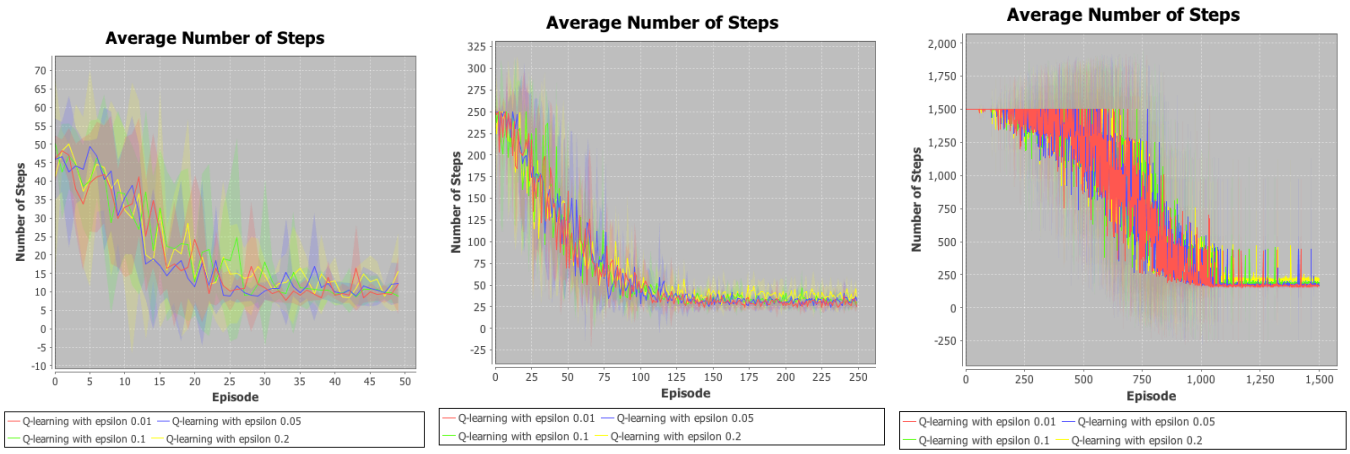

Figure 5 a, b, c: Q-Learning Impact of ε on Average Number of Steps/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

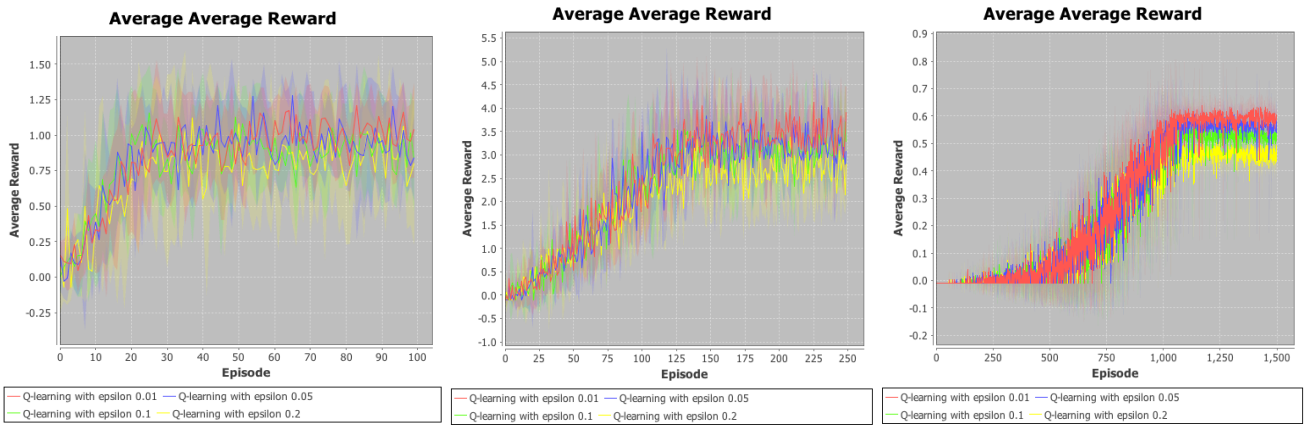

Figure 6 a, b, c: Q-Learning Impact of ε on Average Average Rewards/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

In both Four Rooms and Maze, smaller ε has generally led to lesser number of steps/episode. In fact, the difference is more pronounced in Maze due to its large number of states. In case of the Simple Grid World, the difference is negligible and ε doesn't impact it. In terms of rewards, we observe for Four Rooms and Maze that smaller ε leads to higher average rewards/episode with the difference being very prominent in case of the Maze. The same is generally true for the Simple Grid World although to a smaller extent. A small ε value implies that the agent almost never takes a random action. This makes sense as the Maze is so large that taking random values might mean that the agent may never reach the terminating state; the same is true in case of Four Rooms although to a smaller extent. However, this is not the case for the Simple Grid World. Due to the smaller size of the problem, the agent can do just as well by exploring the space rather than exploitation. However, large problems like the Maze rely on exploitation to converge faster. Similarly, we now observe the effect of varying the learning rate $\propto$ on average number of steps to convergence as shown in Figures 7, 8.

The one with an obvious trend was the Maze, which converged in only 300 episodes with $\propto$ = 0.7, in 450 episodes with $\propto$ = 0.4, in 1000 episodes with $\propto$ = 0.2 while $\propto$ = 0.1 converged a lot later in 1500 episodes. It follows a similar trend in terms of average rewards/episode. Likewise, Four Rooms had a similar pattern in both aspects although the difference is less prominent. Lastly, Simple Grid World didn't have any particular trend as $\propto$ = 0.1, 0.7 performed slightly better than the $\propto$ = 0.2, 0.4.

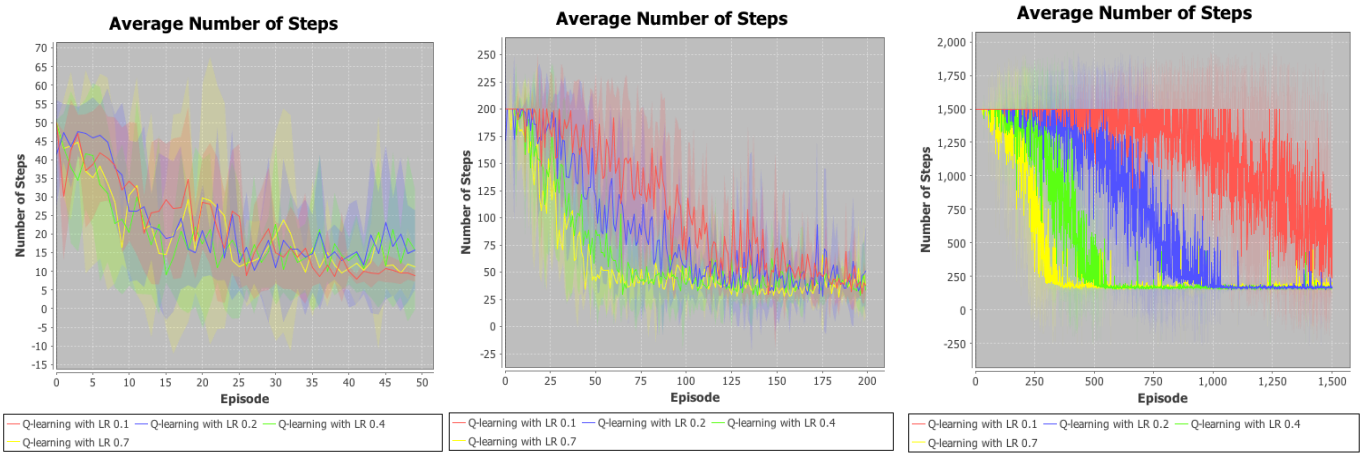

Figure 7 a, b, c: Q-Learning Impact of $\propto$ on Average Number of Steps/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

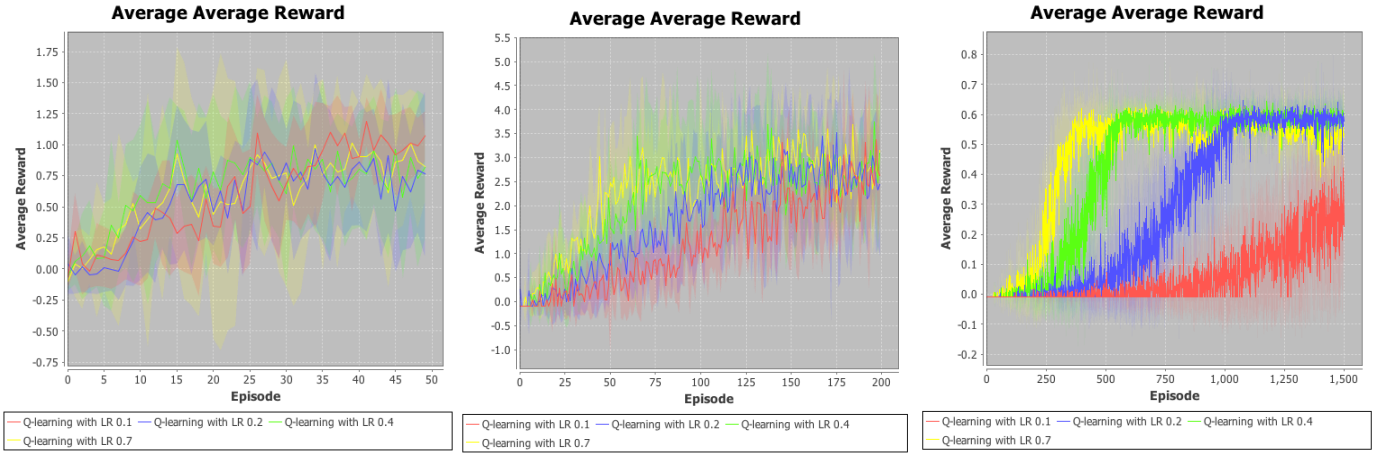

Figure 8 a, b, c: Q-Learning Impact of $\propto$ on Average Average Rewards/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

A preference of large $\propto$ in the Maze and Four Rooms implies that the agent continuously relies on new information while this is less important in toy problems such as the Simple Grid World.

Next, we look at the impact of varying Q_{init} on the performance of QL as shown in Figures 9, 10. Across all 3 problems, lower values of Q_{init} have converged faster than higher values when it comes to average number of steps per episode. In case of average rewards per episode as well, agents with higher values of Q_{init} start with much lower rewards and converge much later. Although the difference is not significant, it can be for a small domain like Small Grid World since we are considering averages in this case. From the experiments conducted, we conclude that complicated problems with a large number of states and blocking conditions (in the form of walls) such as the Maze tend to perform significantly better with very low ε, low Q_{init}, and high $\propto$ values. This is also

true for a relatively mid-sized problem such as Four Rooms. However, in case of much simpler problems such as Simple Grid World, choosing lower $\propto$ values results in faster convergence.

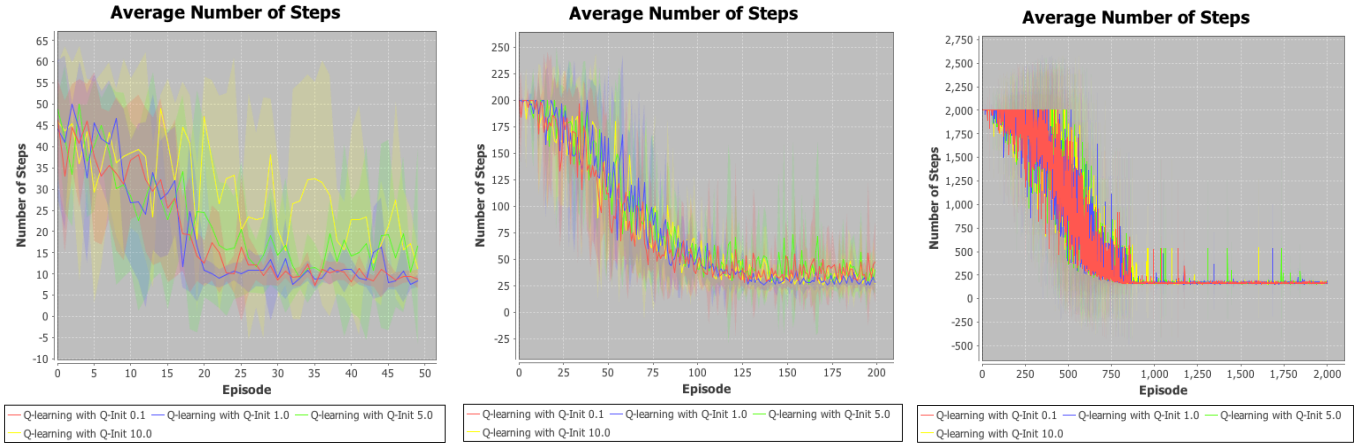

Figure 9 a, b, c: Q-Learning Impact of Q_{init} on Average Number of Steps/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

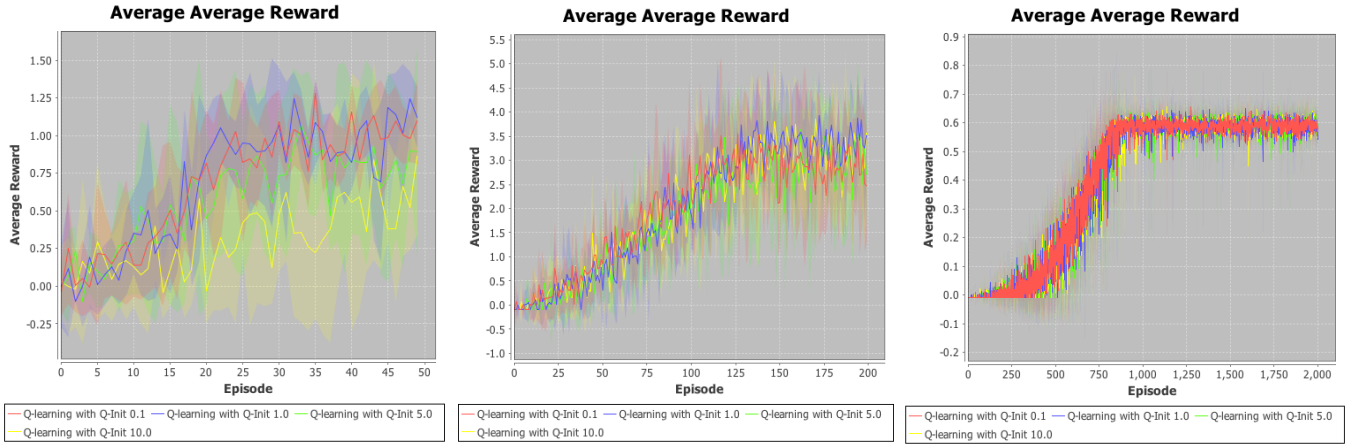

Figure 10 a, b, c: Q-Learning Impact of Q_{init} on Average Average Rewards/Episode on Simple Grid World (a), Four Rooms (b) and Maze (c)

The policy maps obtained via QL across the 3 problems is as shown in Figure 11. The one that clearly strikes out is the one for Maze: there is clear change in gradient from red to blue running in the path directly form the starting point until the ending point. Therefore, while VI and PI many several times never converged in case of the Maze, QL performs reasonably well on it.

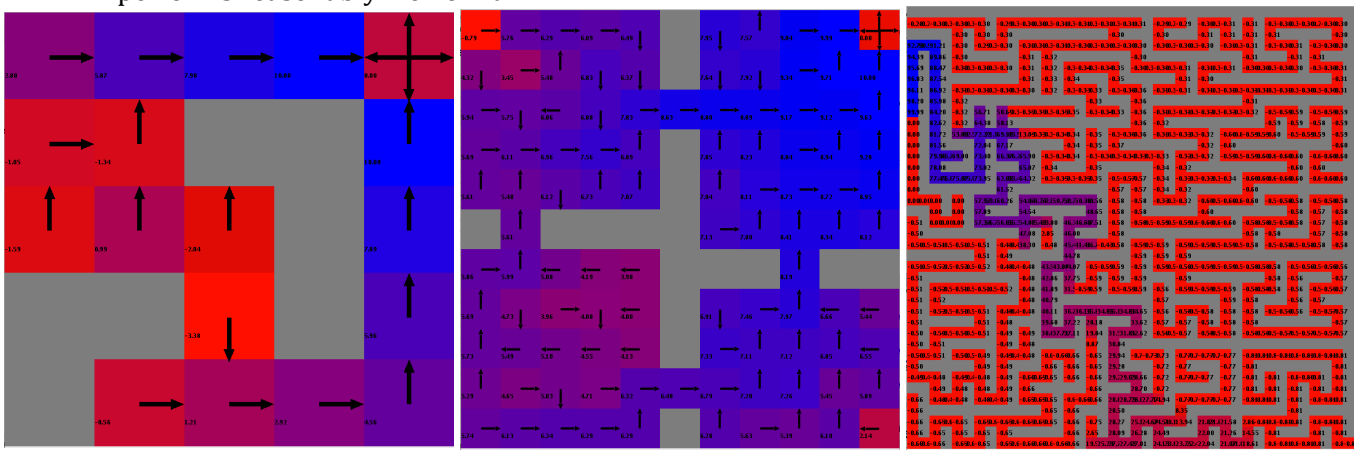

Figure 11 a, b, c: Q-Learning on Simple Grid World (a), Four Rooms (b) and Maze (c)

Similarly, the policy map obtained in case of Four Rooms show a pretty well defined path from the left bottom corner of the room to the terminating state at the top right corner. Across all MDP problems discussed, the strength of Q-learning over value/policy iteration is very evident in case of Maze. We see Q-learning is able to converge in just less than 1000 episodes with reasonable number of steps. However, as discussed earlier in the paper, policy/value iteration is sometimes not able to converge in case of the Maze even in 99,999 steps.

In order to make a comparison between all 3 algorithms explored in this paper, Table 5 summarizes the performance of each on all 3 problems using the best parameters obtained so far with each but the same TR and SR to make a fair comparison between average rewards. The highlighted sections are the highest average reward gained by the agent. Without a doubt, QL performed the best in case of the Maze. This is also evident from the policy map obtained as shown in Figure 11 in comparison to the ones obtained by VI and PI. In case of Four Rooms, VI performed the best in case of Four Rooms while PI came very close, just a tad 0.005 average reward behind VI. VI and PI have performed quite similar in case of SGW and Four Rooms; however, the number of iterations taken by PI is a lot lower than that of VI albeit a higher runtime. Finally, QL performed the best in case of SGW. However, as it is clear from the results, runtime of both QL and PI is usually very high, especially when the number of states is high such as the Maze. In problems of large size, PI and QL may not be feasible.

		SGW	**Four Rooms**	**Maze**
Value Iteration	Steps to Exit	7	23	162
	Runtime	65	162	1587
	Avg. Reward	0.542	0.339	0.509
Policy Iteration	Steps to Exit	4	23	164
	Runtime	15	346	3955
	Avg. Reward	0.375	0.334	0.510
Q-Learning	Steps to Exit	7	26	235
	Runtime	430	536	3641
	Avg. Reward	0.571	0.288	0.621

Table 5: Summary of performance of VI, PI and QL on SGW, Four Rooms and Maze