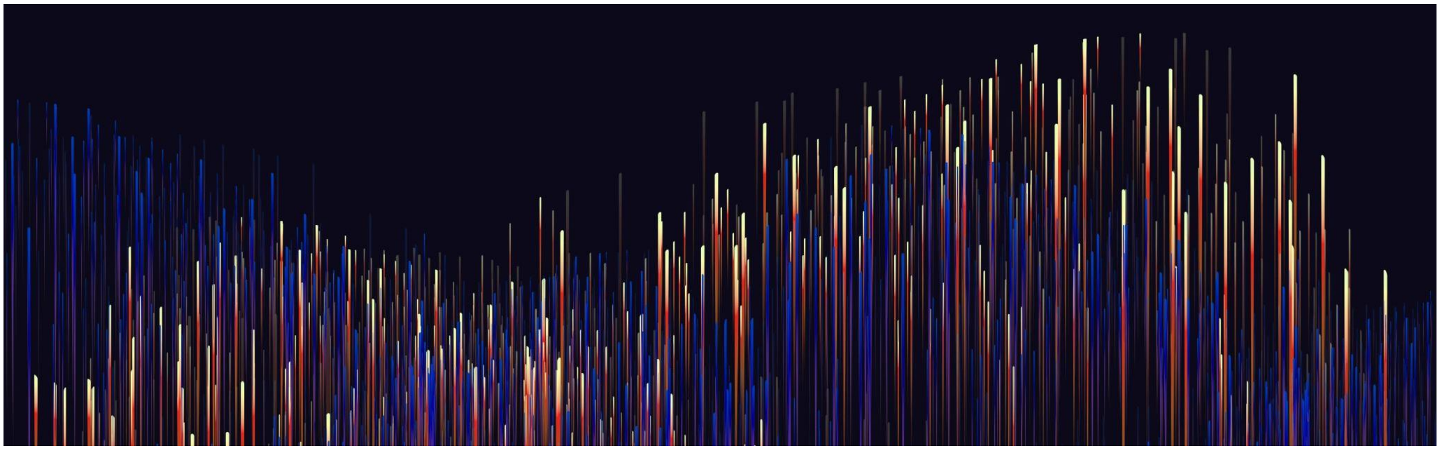


Markov Decision Processes – sabhaee

Introduction:

In this report I attempt to study and explore Markov Decision Process (MDP) by solving two different problems. MDP problems focus on stochastic processes with fixed number of states (s) and actions (a) which for each given action in the space of possible actions evolve from one state to another with a probability that is conditioned only on the current state and the next state (s'). In other words, the transition probability T(s,a,s') from one state to another is blind to all the previous steps and hence has no memory [1]. In this type of problems, the agent's goal is to find a policy for each state that maximizes the final rewards.

To have diverse type of problems and provide breadth to the analysis the two problems with varying size (small and large) and type (Grid world and non-grid world) were selected for this analysis. Each problem will be solved using 3 different methods. First the Value iteration and policy iteration method will be used to solve the problem. These methods are model based which means we all the environment is known to agent by having access to the complete transition probability and reward matrix. Finally, Q learning method will be used to solve the problems. Q learning is a model free reinforcement learning algorithm where the agent starts completely unaware of the transition probability and rewards and only receive information about the environment when it attempts an action and receive a reward and move to a new state.

MDP Problems:

Small/Grid World: The 4x4 Frozen Lake problem from OpenAI gym environment was selected [2]. In this problem we have start location (top left corner) and a goal location (bottom right corner) additionally there are locations on the grid that represent holes. The agent's goal is to reach the goal and while avoiding falling into the holes. Possible actions are as follows {'Left': 0,'Down': 1,'Right': 2,'Up': 3}. Additionally, the environment is slippery (stochastic) which means the chosen action does not necessarily result in the expected direction. This problem poises a very interesting situation where the agent receives a reward of 1 only if it reaches the goal and gets a reward of zero for every other state and action. This is especially challenging in the case of model free methods and incentivizes exploration while requiring lots of iteration. OpenAI documentation considers the problem solved when the policy reaches the goal state more than %78 of times.

Large/Non-grid World: A 400 state Forest management problem from mdptoolbox was selected for the second problem. In this problem each state represents an age range of the forest and at each state the agent needs to decide whether to cut (a=1) or wait(a=0) agent receive a reward of 4 if the agent in the state 399 and it chose wait and a reward of 2 if select cut at this final state. Any time the agent selects the cut before the final state it will result in a reward of 1 and reset the forest to s=0, while selection of wait in any state other that result in no immediate reward. There is also a probability of fire at each state (p=0.1) which if it occurs, will reset the environment to state zero with no rewards. The reward structure of this problem poses a very interesting situation where the agent needs to decide at each state whether to choose the future reward and wait and risk catching fire with the hope to get to final state and receive a higher reward or attempt a cut action to ensure an immediate reward.

In addition to the problem parameters including the rewards and transition probability as described for each MDP above, we need to also study the effect of discount factor, gamma (γ) for each problem. The discount factor represents how much an agent values the future rewards where higher gamma value represents valuing later rewards and vice versa. Different gamma values will result in different optimal policies and in result change the problem solution. The effect of different values of gamma on convergence will be studied for each problem.

Solving Problems:

Model Based Approaches:

Value Iteration:

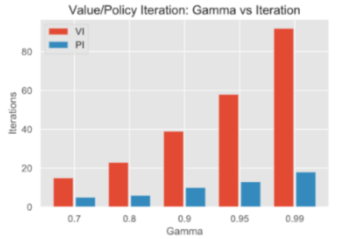
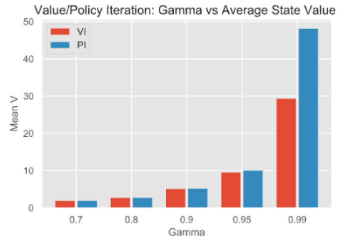
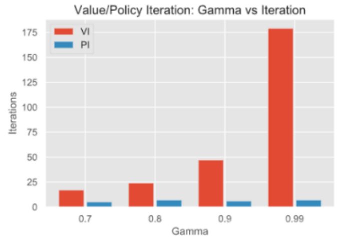
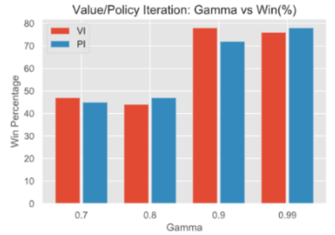
Value iteration algorithm attempt to estimate the optimal state values by iteratively updating the value matrix(V) using the Bellman equation until it converges. The optimal state value is the sum of all the future discounted reward that agent can expect at each state. The convergence is achieved when the difference between the state values at each iteration falls below a threshold (ϵ).

Policy Iteration:

In the Policy iteration algorithm, we start with guess for our initial policy and then iteratively update the value function using the bellman equation and improve our policy to find the actions that maximize the expected values. The algorithm converges when the policy stops changing.

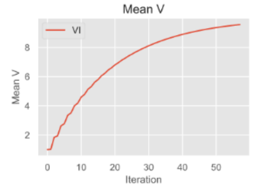
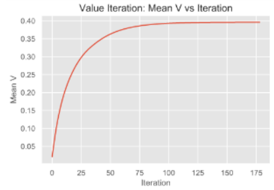
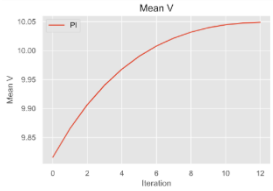
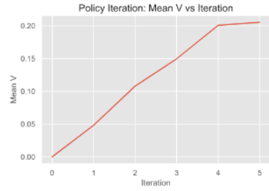
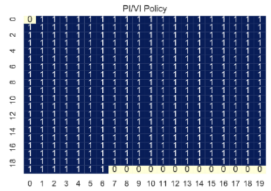
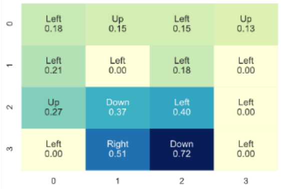
Results:

I first look at the effect of gamma on performance both algorithm and on both problems. Figures below show the result for various values of gammas. As it is clear from all figures in both problems a higher value of gamma result in a higher reward/performance since the agent is valuing the future reward higher than the immediate rewards. At the same time this means the agent requires much more iteration to reach convergence. (Note iterations represented for PI and VI are not the same type of iterations for VI each iteration is s value iteration update and for PI each iteration is a policy iteration. They are plotted on the same graph for brevity.)

Forest Management		Frozen Lake	
 *Figure 1*	 *Figure 2*	 *Figure 3*	 *Figure 4*

Next, we use the discount factor that resulted in the highest state value and performance for both problems to study the convergence behavior and final policy. We can observe that the average value of the V matrix (state value) is converging, and the curves stop when the model converges. Optimal policies for both PI and VI method converge to the same result for both problems. This is expected since in the model-based methods such as PI and VI we know the states and after sufficient iteration the algorithm is guaranteed to find the optimum policy.

In the frozen lake problem, the policy wins 85% of the episodes while the VI converges in 179 value iteration and PI in 6 policy iterations.

Forest Management (gamma=0.99)	Frozen Lake (gamma=0.99)
 *Figure 5*	 *Figure 8*
 *Figure 6*	 *Figure 9*
 *Figure 7*	 *Figure 10*

Reinforcement Learning Method: Q-Learning

Next, I attempt to solve the same problems using Q-learning. This is a model free method where algorithm starts without any knowledge of the rewards and transition probabilities.

It has been proven that Q learning is limit guaranteed to achieve the optimal policy but without proper parameter tuning the number of episodes/ iterations required might be intractable. Before attempting to explore the q learning parameters I first study the effect of the discount factor on each problem but this time using the q-learning method. For this purpose, a set of parameters for each problem was selected. To avoid unreasonable run time a constant number of iteration/episodes were used to study performance given different parameters. For the forest problem this number was selected after experimenting with different sets of parameters to find a visual upper bound for number of iterations that ensure convergence of Q matrix error between iterations. For the Frozen Lake since the objective is to maximize the average score, the running average of cumulative reward over a span of 100 episodes was selected as the performance measure. We can observe that again it is the highest gamma selected that shows higher performance than others (To get an idea of performance for the Frozen Lake the final percentage of win (score value) after applying the policy is plotted and for the Forest the average value of the Q_max matrix which represent the values corresponding the final policy).[Figure 11-14]

Next using the gamma=0.99 (similar to the PI and VI section) hyperparameter analysis was performed. Similar to the gamma study some rule was selected to evaluate the algorithm, for the frozen lake the number of training episode that resulted in the running average of win ratio greater than 0.78 was plotted. In this scenario the algorithm was run until the average last 100 consecutive episodes resulted to an average reward of 0.78 (Solving threshold by OpenAI standard). For the forest the Mean value of the Q_max matrix was plotted.

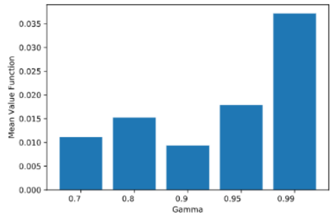
<table>
<tr><th colspan="2">Forest Management</th><th colspan="2">Frozen Lake</th></tr>
<tr><td colspan="2">alpha=0.2, alpha_decay=0.99, epsilon=1, epsilon_min=0.2, epsilon_decay=0.9, n_iter=10000</td><td colspan="2">alpha=0.8, eps=1, numTrainingEpisodes=5000, numTrainingSteps=300, min_epsilon=0, epsilon_decay=0.99</td></tr>
<tr>
<td>


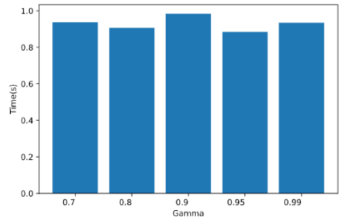
Figure 11
</td>
<td>


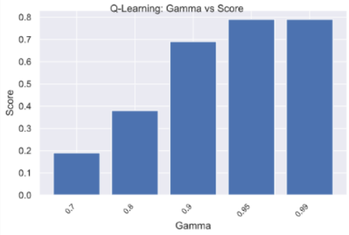
Figure 12
</td>
<td>


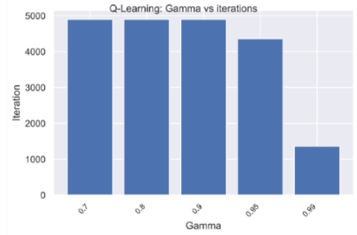
Figure 13
</td>
<td>


Figure 14
</td>
</tr>
</table>

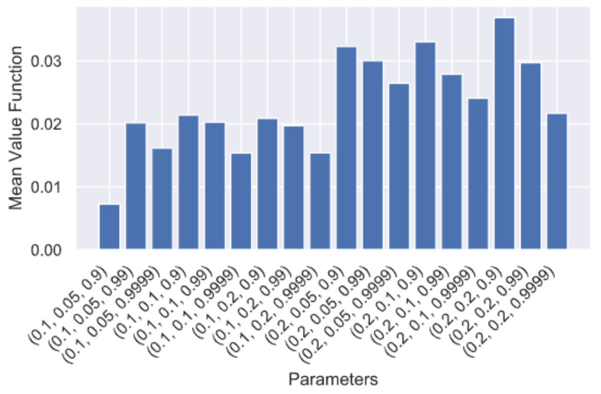


Figure 15 – Forest: Mean V – parameters(apha,min_epsilon,epsilon_decay)

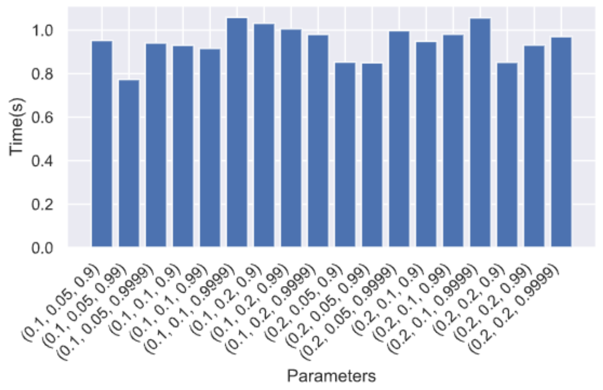


Figure 16: Time to train- parameters(apha,min_epsilon,epsilon_decay)

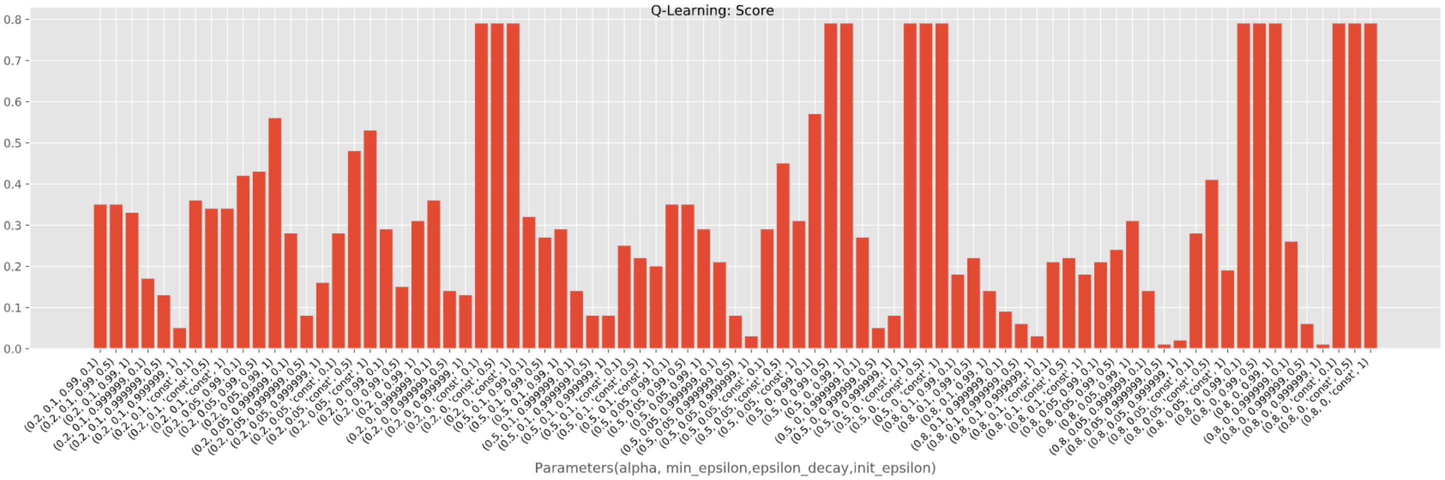


Figure 17 – Frozen Lake: Final Avg score

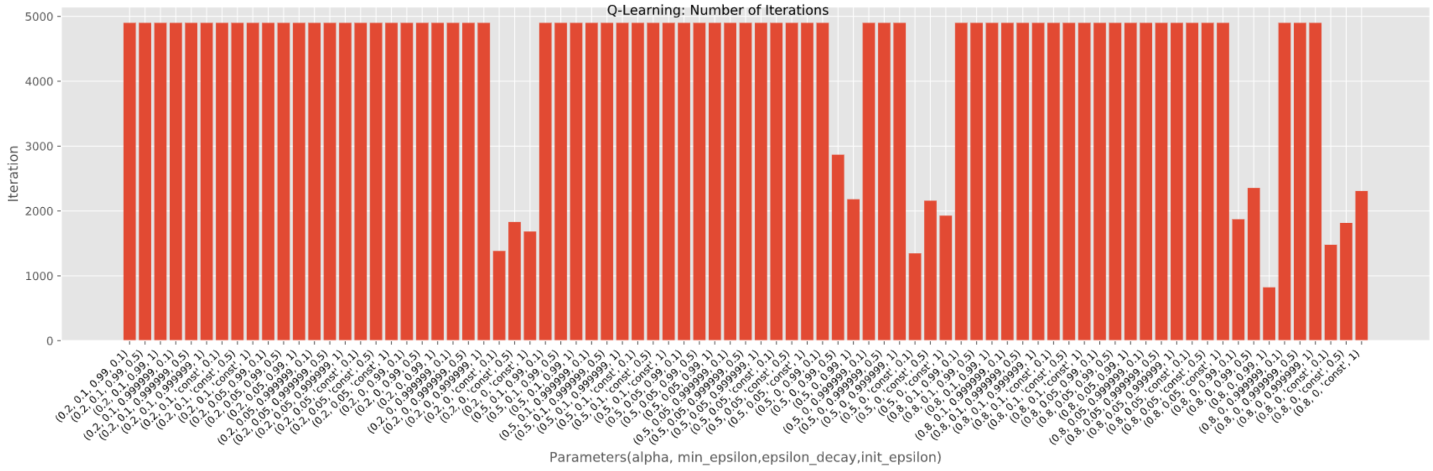


Figure 18- Frozen Lake: Number of Episodes to winning score

Q-Learning HP analysis:

As shown on Figures 15-18 the following parameters were studied for both problems.

Learning rate(alpha): this parameter controls the rate of learning and adjusts the weight given to new experiences comparing to previous ones. A high alpha means we believe the new experiences more strongly and quickly adopt to them while a lower alpha represents skepticism to the new experiences and holing on to the previous values of Q for a longer time.

Epsilon: The success of the Q learning algorithm depends strongly on the tradeoff between exploration vs exploitation. In this context a fully exploitative model relies only on argmax(Q) as its policy for selecting the next state. This approach does not allow the learning model to explore new states and tends to not converge. To solve this issue, we need to introduce some randomness to the action selection, so the model has a possibility to explore. For this purpose, we allow the model to select a random action with the probability of epsilon but at the same time as learning improves, we need to decay this randomness using the epsilon decay. Additionally, to ensure the model still has room for exploration we can leave a minimum epsilon probability for random action.

For the forest problem we can see a set of parameters of alpha=0.2, minimum epsilon=0.2 and decay rate of 0.9 result in higher expected future reward and lower train time. With these parameters the learning rate is low which means the previous values of Q is more strongly preserved while the decay rate quickly takes the algorithm to the base exploration or min_epsilon, and since this value is high it suggests the algorithm will stay explorative for the rest of training.

For the case of the Frozen Lake problem as mentioned earlier due to special form of reward shaping in this problem the learner does not receive any feedback from environment unless it reaches the goal state. For this reason, it is expected that a explorative model should perform better. Studying Figure 17 we can see that only a handful of the parameter sets reach the threshold of 0.78, and among these set of (alpha=0.8, min_epsilon=0, epsilon_decay=0.99, initial_epsilon=1) reaches this solving state in the smallest number of episodes (less than 1000 episodes). Let's unpack these parameters, a high alpha shows a quick learning rate updating the Q values more confidently while initial-epsilon of 1 means the learner starts with completely random actions and then decay it all the way to zero an becoming completely deterministic.

Q-Learning model:

Now we use the best parameters from the hyperparameter analysis to solve the problems.

Based on figures 19 and 20 for the forest problem it appears the learner did not converge even after 150000 iterations

Forest Management

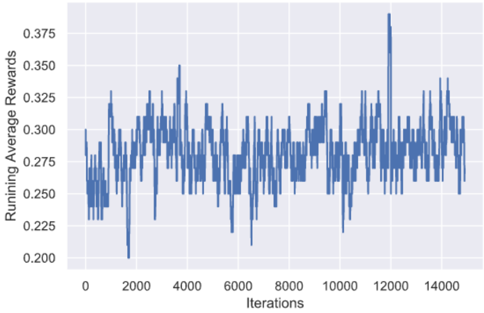
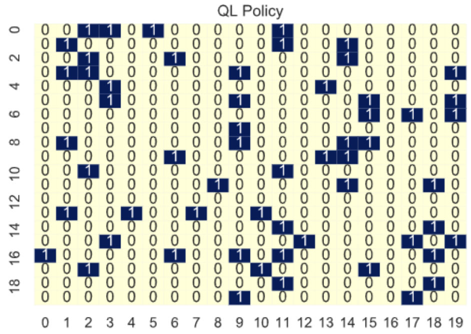


Figure 19



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
4	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0
10	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1
16	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0
	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Figure 20

Frozen Lake

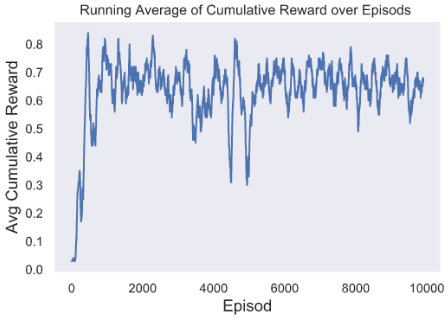
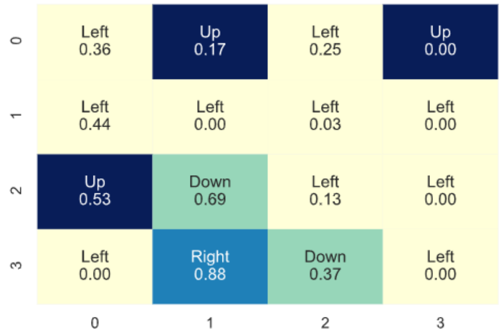


Figure 21



0	Left 0.36	Up 0.17	Left 0.25	Up 0.00
1	Left 0.44	Left 0.00	Left 0.03	Left 0.00
2	Up 0.53	Down 0.69	Left 0.13	Left 0.00
3	Left 0.00	Right 0.88	Down 0.37	Left 0.00
	0	1	2	3

Figure 22

and as it is clear the policy does not match the optimum policy achieved by the value and policy iteration. This non-convergence can be associated to the size of the problem and might suggest the requirement of more iterations, also it is another testament to the effect of reward shaping in this problem where the maximum rewards only reaches at the end while due to the size of the problem learner might not reach that stage since it attempts a cut or get subjected to the fire and need to start over. Figure below shows the number of iterations increased from 150000 to 20x10^6:

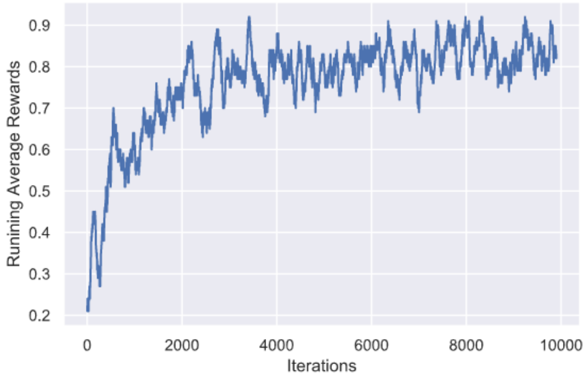
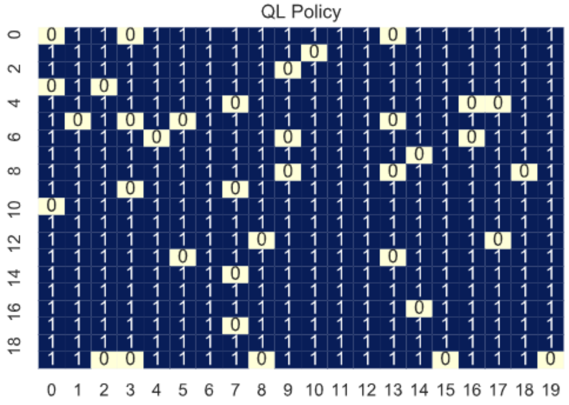


Figure 23



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	1	1
	1	0	1	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
6	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1
	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
10	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1
	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
14	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	0

Figure 24

We can now see the algorithm is starting to converge (data was recorded every 200 iterations hence the 10000x200=20MIL). We can also how the policy is converging to the optimal policy for the PI/VI method with more cut actions to expedite the immediate reward and avoid ever losing everything due to fire.

On the other hand, based on Figures 21 and 22, Frozen Lake problem shows a convergence after about 6000 episodes and the policy achieved, matches the ones obtained by the value and policy iteration. One interesting aspect of the cumulative reward graph in Figure 21, we can see a large drop and spike around episode 5000. This situation can be explained as a failure in exploration which adversely affected the learner's success, but eventually the learner converges by going through more cycles of training.

Problem complexity: State-space size

Although the previous two problems were not similar in nature and a direct comparison is not possible, we still saw the convergence issue for problem with the larger size. To study the effect of size complexity further we look at the different sizes of the same problems. For start I consider an 8x8 grid Frozen Lake which has 64 states a 4X increase in the problem size. Figures 25-28 below show the policy and convergence curves for value and policy iteration:

We can see that this time both algorithms need more iteration to converge (VI: 201 and PI:10). Interestingly this time the policies do not match. Considering the stochastic nature of the problem it seems as the size of problem is increasing it is less likely for the PI and VI to converge to the same solution, but at the same time due to the probabilistic nature of problem there is not one single policy that solve the environment. We can further explore the two sizes of problem by looking at the average reward and time to converge:

The train time for both methods has increased. On the other hand, the average reward (success) for the PI is now less than the VI which was the same for the case of smaller problem.

Value Iteration

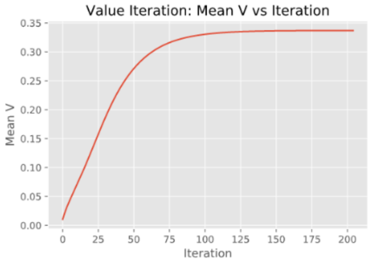
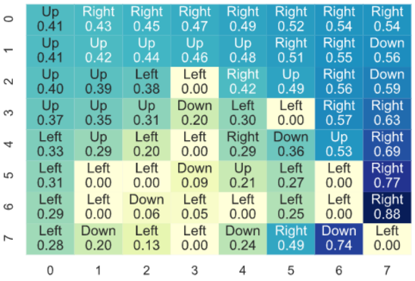


Figure 25



	0	1	2	3	4	5	6	7
0	Up 0.41	Right 0.43	Right 0.45	Right 0.47	Right 0.49	Right 0.52	Right 0.54	Right 0.54
1	Up 0.41	Up 0.42	Up 0.44	Up 0.46	Up 0.48	Right 0.51	Right 0.55	Down 0.56
2	Up 0.40	Up 0.39	Left 0.38	Left 0.00	Right 0.42	Up 0.49	Right 0.56	Down 0.59
3	Up 0.37	Up 0.35	Up 0.31	Down 0.20	Left 0.30	Left 0.00	Right 0.57	Right 0.63
4	Left 0.33	Up 0.29	Left 0.20	Left 0.00	Right 0.29	Down 0.36	Up 0.53	Right 0.69
5	Left 0.31	Left 0.00	Left 0.00	Down 0.09	Up 0.21	Left 0.27	Left 0.00	Right 0.77
6	Left 0.29	Left 0.00	Down 0.06	Left 0.05	Left 0.00	Left 0.25	Left 0.00	Right 0.88
7	Left 0.28	Down 0.20	Left 0.13	Left 0.00	Down 0.24	Right 0.49	Down 0.74	Left 0.00

Figure 26 – VI-Policy

Policy Iteration

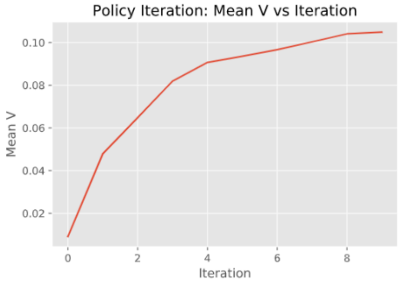
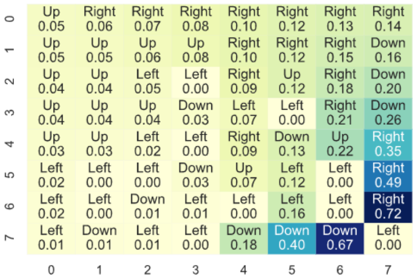


Figure 27



	0	1	2	3	4	5	6	7
0	Up 0.05	Right 0.06	Right 0.07	Right 0.08	Right 0.10	Right 0.12	Right 0.13	Right 0.14
1	Up 0.05	Up 0.05	Up 0.06	Up 0.08	Right 0.10	Right 0.12	Right 0.15	Down 0.16
2	Up 0.04	Up 0.04	Left 0.05	Left 0.00	Right 0.09	Up 0.12	Right 0.18	Down 0.20
3	Up 0.04	Up 0.04	Up 0.04	Down 0.03	Left 0.07	Left 0.00	Right 0.21	Down 0.26
4	Up 0.03	Up 0.03	Left 0.02	Left 0.00	Right 0.09	Down 0.13	Up 0.22	Right 0.35
5	Left 0.02	Left 0.00	Left 0.00	Down 0.03	Up 0.07	Left 0.12	Left 0.00	Right 0.49
6	Left 0.02	Left 0.00	Down 0.01	Left 0.01	Left 0.00	Left 0.16	Left 0.00	Right 0.72
7	Left 0.01	Down 0.01	Left 0.01	Left 0.00	Down 0.18	Down 0.40	Down 0.67	Left 0.00

Figure 28

Frozen Lake (4x4)

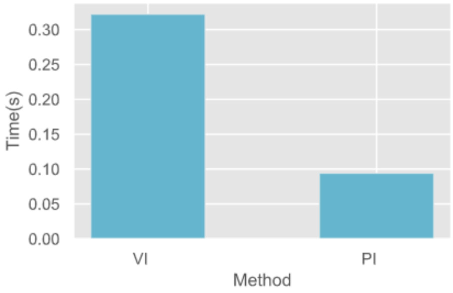


Figure 29 – Time to converge(4x4)

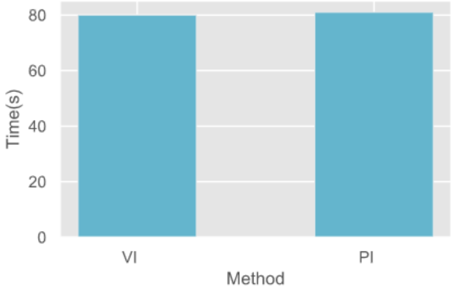


Figure 30 – Average rewards(4x4)

Frozen Lake (8x8)

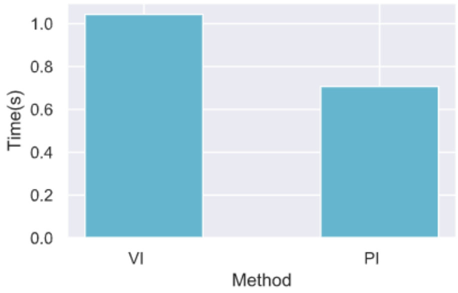


Figure 31 – Time to converge(8x8)

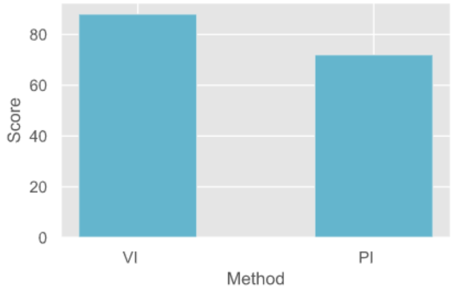


Figure 32- Average rewards(8x8)

References:

1. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, Aurélien Géron, ISBN: 9781492032649
2. https://gym.openai.com/envs/FrozenLake-v0/