CS-7641 Machine Learning: Assignment 4, Markov Decision Processes

Pavel Ponomarev
pavponn@gatech.edu

1 INTRODUCTION

In this project, we [1] explore Markov decision processes (MDPs) and various algorithms to solve them. The main goal is to gain more experience with a machine learning setting where agent can interact with a world more directly and apply reinforcement learning techniques studied.

We analyse performance and outputs of algorithms such as *Value Iteration (VI)*, *Policy Iteration (PI)*, and *Q-Learning (QL)* on two different MDPs. Let us overview their main features of these methods. Value iteration is a dynamic programming algorithm for solving Markov decision processes (MDPs). It works by iteratively refining the value function, which is the expected cumulative future reward for each state in the MDP. Once the value function converges, it is then used to compute the estimated optimal policy, which is the sequence of actions that maximizes the expected cumulative future reward. Assuming finite state and action space, as well as bounded reward, the algorithm is guaranteed to converge to an optimal policy.

Policy iteration is another dynamic programming algorithm for solving MDPs. It works by iteratively improving a policy. The policy is improved by evaluating the current policy and then updating it to be greedy with respect to the value function. Once policy does not change, the algorithm is considered to converge. This method is also guaranteed to converge to the optimal policy for any finite MDP.

Q-Learning, a model-free reinforcement learning algorithm, transcends the limitations of dynamic programming by learning directly from interactions with the environment. This off-policy algorithm maintains a Q-value function, representing the expected cumulative reward for taking a specific action in a given state. Q-Learning explores the environment, updating Q-values based on observed rewards and transitions. Note that unlike Value and Policy iterations, this algorithm does not have access to transition probability and reward matrix. Convergence and the optimality of the resulting policy heavily depend on the algorithm's hyperparameters.

2 MDP PROBLEMS OVERVIEW

In this section, let us give a quick description of the chosen problems (Section 2.1 and Section 2.2) and then explain their potential interesting properties for the analysis and hypothesise the algorithms' behaviour (Section 2.3).

2.1 Forest Management

The first Markov Decision Process problem we use in this work is *Forest Management*. The problem has S states $\{0, 1, \ldots, S-1\}$ that represent the age of the forest with $S-1$ being the oldest. The agent has two actions available to manage the forest: *Wait (W)* and *Cut (C)*. An action is decided each year with first the objective to maintain an old forest for wildlife and second to make money selling cut wood. Each year there is a probability p that a fire burns the forest. The reward structure for managing the forest depends on the forest's age. If the forest is cut when it is in its oldest state, the reward is 2. On the other hand, if the forest is not cut in its oldest state, the reward is 4. If an agent chooses to perform a Cut action on a forest of 0 age, the reward for this action is 0. Additionally, cutting the forest in any state other than the oldest or youngest results in a reward of 1, while reward for waiting in any state other than the oldest is 0. In this work, we fix the probability of the fire being 0.1 at any possible state and mainly focus on the problem with the state size $S = 100$, however we do explore other world sizes for comparison. We also analyse how the problem changes depending on the value of the future reward (discount factor γ). With respect to the assignment requirements, this problem is considered *small*.

2.2 Taxi

Another MDP problem we focus on in this assignment is *Taxi*. The world represents an $n \times n$ grid where an agent needs to operate as a taxi driver. Their goal is to pickup a passenger and then drive to the passenger's destination and

1 The work described in this report is completed fully independently by one sole author (Pavel Ponomarev). The pronounce "we" is used to follow academic writing style.

drop them off there. A passenger's initial location as well as their drop off point might be in one of the four locations indicated on the grid: *Red (R)*, *Green (G)*, *Yellow (Y)* and *Blue (B)*. In between some of the grid cells, there are walls that prevent taxi moving directly between neighbouring locations. In total, agent can be in $n \times n \times 5 \times 4$ states ($n \times n$ taxi locations, 5 possible locations of the passenger including the case when the passenger is in the taxi and 4 possible destinations). The agent has 6 actions available: move *South*, *North*, *East*, *West*, *Pickup passenger* and *Drop off passenger*. In terms of reward, an agent gets $+20$ for delivering the passenger, -10 for executing pickup and drop-off actions illegally and -1 in any other state. In this work, we mainly analyse the behaviour of reinforcement techniques on the problem of the state size 2000 ($n = 10$), however we do explore other world sizes for comparison. Similarly, we analyse how the algorithms' performance changes depending on the discount factor γ. In terms of the assignment requirements, we consider this problem being *large*.

2.3 Problem properties & Hypotheses

The Forest Management problem is relatively simple as it has only two actions, but yet has some interesting properties. Mainly, the stochastic world should require an agent to weight the pros and cons of chasing the highest reward depending on how "far" the current state is from the oldest state of the forest. Given the problem description, we believe that both value and policy iteration algorithms should converge to the same optimal policy. Even more, there is only one optimal policy, that can be written as follows $\pi(0) = W$ and $\forall s_i >= s_t : \pi(s_i) = W$, otherwise $\pi(s) = C$. Here s_t is the smallest state such that the expected value for taking a risk and waiting with a hope to reach the reward from the oldest state is more reasonable then collecting a guaranteed reward now but minimising the chances for higher reward in the future. With the Q-Learning approach, the problem would require to find the right trade-off between exploration and exploitation: since the agent does not have information about all transitions and rewards it is important to make sure that it explores waiting actions in states closest to the oldest and does not overvalue waiting actions in "younge" states especially when discount factor is high.

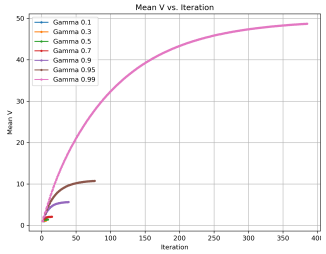
The Taxi problem is different though: while it does not involve stochastic process (an action executed in a given state always yields the same result), the problem space is generally much bigger even for relatively small grid size n (e.g., if $n = 3$, number of states $S = 320$, for $n = 10$, $S = 1000$). Both value and policy iteration algorithms will converge to an optimal policy. However, while both policies should be maximising the expected reward from a given state, we do not expect these methods to converge to the same policy. Indeed, one can expect that policy where some actions might be altered (e.g., move South and move West) have the same projected reward. At the same time, Q-Learning algorithm does not know about deterministic nature of transitions anyway, so it should not help this method. At the same time, given that the main problem size we explore ($S = 2000$) is very big, it is crucial to ensure that the agent can explore all states. At the same time, high exploration might prevent the protocol from finding the optimal actions and converging to an optimal policy.

3 TECHNOLOGIES

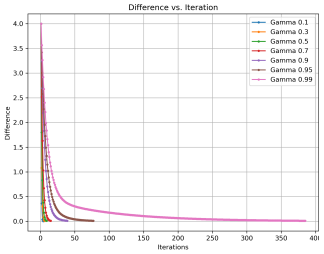
We choose Python 3 as an implementation language for this assignment. We utilise libraries such as `hiive-mdptoolbox`[2] and `gym`[3]. More specifically, we take the Forest Management problem as well as algorithm implementations from `hiive-mdptoolbox`. Additionally, we need to add some functionalities (to enable logging extra statistics) to Policy Iteration and Q-Learning implementations and thus our code contains custom versions of these algorithms. From `gym`, we take the implementation of the Taxi problem. However, their implementation only provides maps of sizes 5×5, which made us build on top additional functionalities: we introduced maps of other sizes from 3×3 to 10×10, enabling support of these maps in the environment.

4 VALUE ITERATION

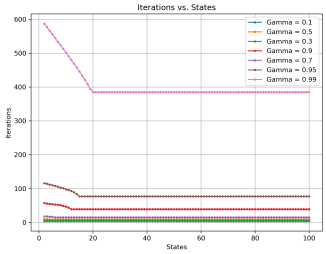
To evaluate convergence of the Value Iteration algorithm, we will rely on metrics like *Mean V*, and *Mean V Difference (Error)* between algorithm iterations. For both problems, we set the maximum number of iterations to be 1000. Algorithm might stop earlier if at some point the maximum (by element) difference between V_i and V_{i-1} falls below $\epsilon = 10^{-2}$. We also analyse clock wall times and number of iterations algorithms perform depending on the problem size and explore policies.



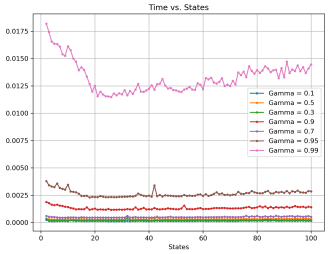
(a) Mean V vs Iterations, size=100



(b) Error vs Iterations, size=100



(c) Iterations vs Problem state size



(d) Time vs Problem state size

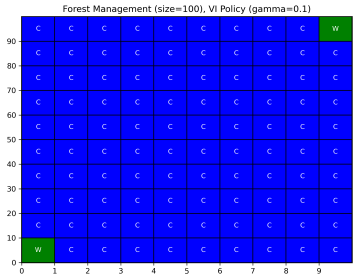
Figure 1—Forest Management with Value Iteration: Statistics

4.1 Forest Management

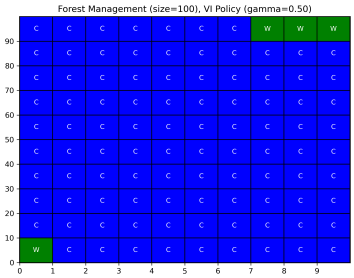
We depict the results of running the Value Iteration algorithm on the Forest Management MDP in Figure 1. As we can see from Figure 1a and Figure 1b, the algorithm converged for all discount factors γ. Interestingly, we notice that higher values of gamma result in more iterations (the same conclusion can be done by looking at Figure 1c). We believe that this makes sense in theory. Value iteration proceeds while updated version of the value function is different from the previous one by at least chosen epsilon. With a higher γ the the value function of a given state propagates to another state's value function with a much higher absolute value (or in other words, the reward is distributed further), which then results in bigger changes in value function between iterations compared to smaller values of gamma.

In Figure 1c, the surprising observation emerges: the number of iterations needed by the Value Iteration algorithm remains constant for state sizes exceeding 20, regardless of the chosen gamma. Contrary to intuition, a higher number of states doesn't consistently demand more iterations for convergence, as it hinges on the selected epsilon threshold. The graph implies that iteration values plateau when the value function signal falls below the chosen ϵ for state sizes beyond 20. Interestingly, for smaller states, the algorithm may require more iterations, exemplified by $\gamma = 0.99$ needing about 500 iterations for a state size of 11, while state sizes exceeding 20 only need around 380. This is likely attributed to cyclic propagation of the reward/value function, losing this property with increased state size due to lower probability of such transition chains and the impact of accumulated discount factor.

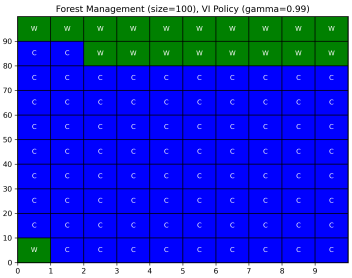
From Figure 1d we can see that the algorithm takes most time for the small problem state sizes: this is due to the higher number of iterations we discussed above. Then after it reaches minimum in certain states (they vary for different values of gamma), the time starts to increase slightly. This is expected given that the number of iterations remains the same and the increase is driven only by the operations on bigger matrices that do not contribute as much given that they are vectorised.



(a) $\gamma = 0.1$



(b) $\gamma = 0.5$



(c) $\gamma = 0.99$

Figure 2—Forest Management (size=100) with Value Iteration: Policies

We visualise obtained policies in Figure 2 for different values of γ: 0.1, 0.5 and 0.99. The age of the forest is visualised as a grid for convenience, the years start in the lower left corner and increase from left to right, from bottom to up.

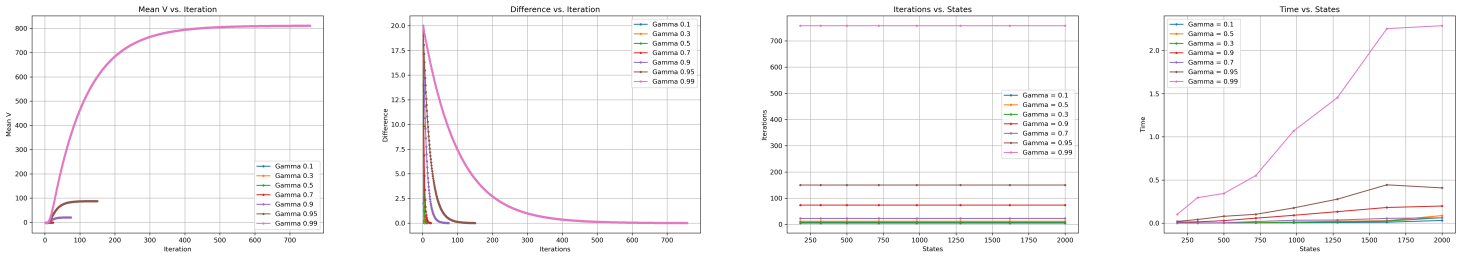
2 https://pypi.org/project/mdptoolbox-hiive/

3 https://www.gymlibrary.dev/index.html

First, we notice that the algorithm does converge to a policy we informally described in Section 2.3. It always performs a Wait action in the state $s = 0$ and $s = 99$ (i.e., $S - 1$), since Cut has a lower immediate or/and future reward. Then with the increase of discount factor, number of states closest to the oldest where agent's preference to risk increases, given that value of future reward grows.

4.2 Taxi

The results of running Value Iteration on Taxi problem are summarised in Figure 3. The algorithm was able to converge for all discount factors γ as we can see in Figure 3a and Figure 3b. We can note that that problems with higher discount rate require more iterations in order to converge, similarly to what we saw in Forest Management problem and, in fact, the same discussion as we had in Section 4.1 would apply here.

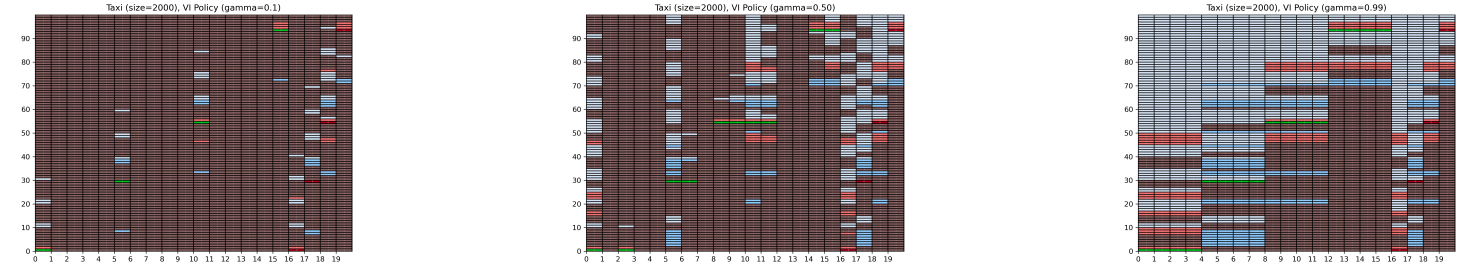


(a) Mean V vs Iterations, size=200 *(b)* Error vs Iterations, size=2000 *(c)* Iterations vs Problem state size *(d)* Time vs Problem state size

Figure 3—Taxi with Value Iteration: Statistics

Similarly to our observations with the Forest Management problem using Value Iteration, we find that the number of iterations in the Taxi environment is independent of the problem's state count. Notably, unlike the Forest Management scenario, we do not observe a higher number of iterations for smaller state sizes in the Taxi environment. This is likely influenced by the fact that the minimum number of states is 180 (for the grid world 3×3), while Forest Management could have around 10-20 states. Such a size is substantial enough to avoid the additional refinement observed in the first MDP with small number of states.

We visualise policy in a grid like manner in Figure 4. X-axis of a cell state represents the combination of destination location and passenger location, Y-axis captures the id of the current location of the taxi. While it might not be possible to fully grasp the meaning of the policy due a big state size, we can clearly see that with the low discount factor agent is barely motivated to move in a direction that would bring it closer to the goal, instead it decides to almost always do the same step, since the reward from the destination state does not propagate well enough (potentially this problem would be less severe with smaller epsilon.) With higher discount rate, agent gets a better understanding of the highest reward location due to "further" propagation.



(a) $\gamma = 0.1$ *(b)* $\gamma = 0.5$ *(c)* $\gamma = 0.99$

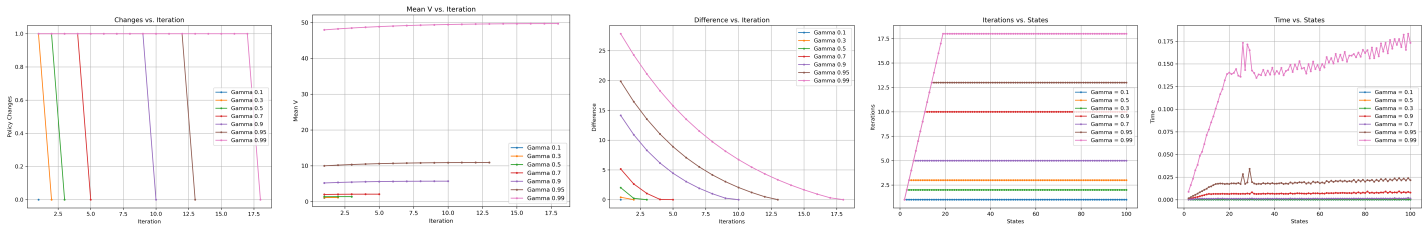
Figure 4—Taxi (size=2000) with Value Iteration: Policies

5 POLICY ITERATION

To understand the convergence of the Policy Iteration algorithm, we will rely on a metric specific to it: *number of policy changes* between iterations. We will also look at the ones we used for Value Iteration to see if there are any major changes. For both problems, we set the maximum number of iterations to be 1000, however we expect that algorithm will stop much earlier after reaching a policy that it could not improve. Similarly to what we did in Section 4, we also

analyse clock wall times and number of iterations algorithms performs depending on the problem size and explore policies.

5.1 Forest Management



(a) Changes vs Iterations, size=100

(b) Mean V vs Iterations, size=100

(c) Error vs Iterations, size=100

(d) Iterations vs Problem state size

(e) Time vs Problem state size

Figure 5—Forest Management with Policy Iteration: Statistics

The summary of the Policy Iteration algorithm run on the Forest Management MDP is presented in Figure 5. According to the number of policy changes (vs iteration) depicted in Figure 5a, the algorithm converges for the problem of size 100 for all gammas. We notice that the Policy Iteration requires significantly less iterations than Value Iteration. Indeed, this is expected given that each iteration of the Policy Iteration algorithm is "more impactful" than an iteration of the Value Iteration. Even more, when VI performs the evaluation of the policy to update it in a given operation, it performs a variation of VI algorithm. This chart also reveals an interesting property of this problem by showing us that on each iteration we update action for exactly one state. Since the algorithm proceeds with the policy that maximises the immediate reward for almost all states s, the action chosen by such a policy should be Cut. The only exceptions are states $s = 0$ (where the immediate reward for performing Cut action is zero) and $s = S - 1$ (where immediate reward from Wait action is higher). Now, let us consider what algorithm does and for simplicity fix $\gamma = 0.99$. On the first iteration, when the method tries to update the policy, it realises that value function can be improved in the state $s = (S-1)-1$ if we change Cut action to Wait. The same happens on the next iteration, but now for state $s = (S-1)-2$. This happens until discount factor diminishes the reward completely.

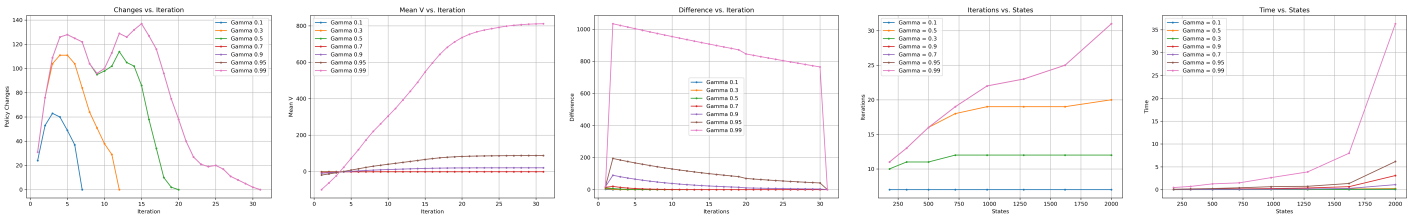
As shown in Figure 5b, the values of Mean V converge to the same vertical asymptotes as with Value Iteration (discussed in Section 4.1). Given our hypothesis that resulting policies should turn out the same (as stated in Section 2.3), it makes sense (and is the case as we mention below). We also notice that the initial Mean V of PI is already much higher than what we noticed using VI. The difference between value function also converges to zero (Figure 5c). Notably, the error change is very smooth, which is most likely caused by the fact that on each iteration the policy makes only one update.

As we can see in Figure 5d, the number of iterations grows with the increase of the state size until a specific states size (different for each gamma) and then remains flat. These "turning point" states are the same as for VI in Figure 1c. This can be explained by the way algorithm proceeds, which we already described when talking about number of policy changes. Up until a certain state, the reward from the oldest stated propagates to the youngest state and the length of this chain only depends on the discount factor (and probability of the fire which is fixed in our analysis). The increase in number of iterations with state size in the beginning leads to a faast increase in time taken by algorithm, after this it slowly increases due to the necesity to perform operations on higher matrices (Figure 5e).

As we hypothesised, the algorithm converges to the same policy as Value Iteration and thus we avoid listing it as a separate figure to save space.

5.2 Taxi

We provide an overview of Policy Iteration algorithm runs on the Taxi problem in Figure 6. Similarly to Forest Management, we notice that the number of iterations required for convergence is significantly smaller for PI compared to VI (Figure 6a). Interestingly, the number of policy changes performed on each iteration first grows very fast, reaching the peaks, fluctuates around the maximum number of changes (for some number of iterations k, higher γ results in higher k) and then number of changes goes down. This is likely related due to the large environment and its structure: in the beginning few elements of the value function can be refined (and thus actions in policies) but on every iteration this number exponentially increases up due to the grid world like problem structure. This happens until a certain point, after which it then starts decreasing similarly due to world structure. The number of iterations depends on γ as it defines how far the value function can be propagated. We also notice that for all discount rates, Mean V converges



(a) Changes vs Iterations, size=2000

(b) Mean V vs Iterations, size=2000

(c) Error vs Iterations, size=2000

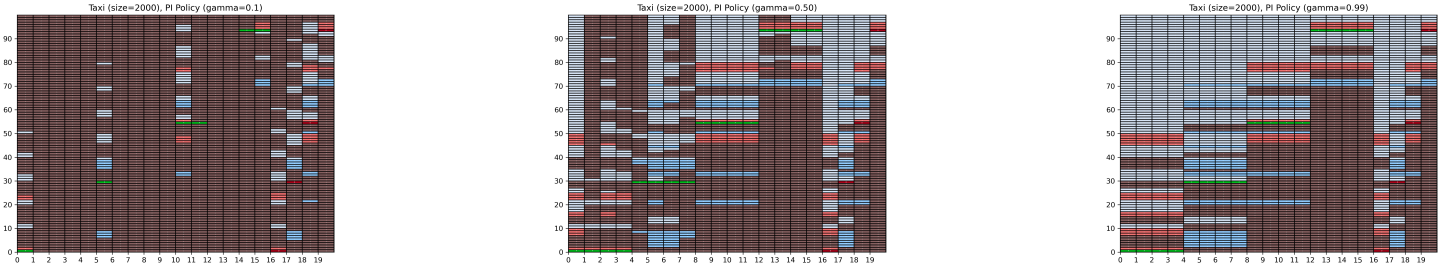
(d) Iterations vs Problem state size

(e) Time vs Problem state size

Figure 6—Taxi with Policy Iteration: Statistics

to the same values as in the case for VI (Figure 6b), suggesting that policies are comparable. From Figure 6c, PI makes much bigger jumps on every iteration compared to VI, making much higher updates to the the value function, which relates well with theory given that each step of Policy Iteration involves estimation of the whole value function. In the end, jump to the values close to zero is expected, since the last iteration does not result in policy update.

In Figure 6d, we find something interesting: whether the number of iterations required for convergences grows (and how fast) with the increase in the number of states depends on the discount factor γ. This happens because algorithm starts with "immediate reward optimal" policy and then tries to improve it taking into account future reward. However, if the problem does not value future reward as much, it does not propagate far away. For example, in case of $\gamma = 0.1$, the reward does not reach furthest states at all, which suggests that it is almost never a good value of a hyperparameter to use if we view Taxi MDP as a problem of finding the shortest path to a passenger and then to its destination. At the same time, gamma values close to 1 might be very beneficial. Additionally, we see that for all gammas the wall clock time increases with the growth of the state size at least due to operations with bigger matrices. For big values of gamma, i.e., $\gamma = 0.9$, the main reason for a much bigger increase is growing number of iterations.



(a) $\gamma = 0.1$

(b) $\gamma = 0.5$

(c) $\gamma = 0.99$

Figure 7—Taxi (size=2000) with Policy Iteration: Policies

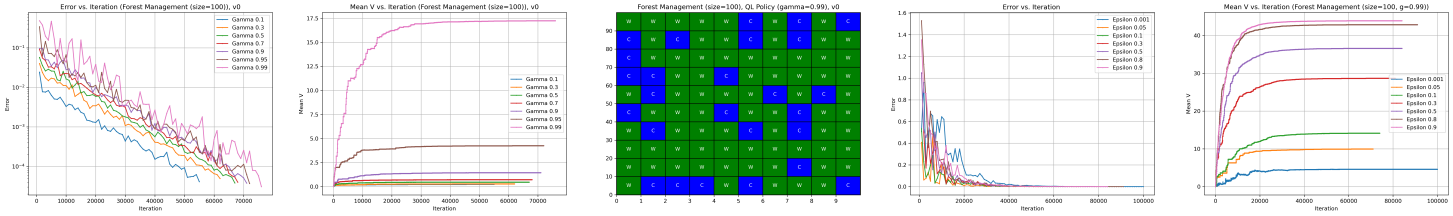
We visualise resulting policies in Figure 7 for values of discount factor 0.1, 0.5 and 0.99. While it is hard to tell from the visual comparison, whether the obtained policies are the same as their corresponding version obtained via VI, we can still see that they follow the same structure. I.e., policies are more eager to seek passenger and their destination for problems with higher discount factor. In order to compare PI and VI policies, we run both of them on the Taxi environment of size 2000, N = 31 episodes with randomly setup initial states and using the total rewards of all episodes to approximate the rate of divergence of two policies. To avoid running forever in cases when policy does not arrive in the terminating state, we truncate each episodes to 100 steps max. The results for each gamma are as follows: $\gamma = 0.1$ — 0.03, $\gamma = 0.3$ — 0.06, $\gamma = 0.5$ — 0.45, $\gamma = 0.7$ — 0.19, $\gamma = 0.9$ — 0.0, $\gamma = 0.95$ — 0.0, and $\gamma = 0.99$ — 0.0. Interestingly, we practically notice that even though both PI and VI converged to the optimal policies, these optimal policies might generate different results for values of gammas below 0.9. This most likely happens due to the different sensitivity of the algorithms (i.e., explicit epsilon in VI and implicit epsilon in PI), which affects difference in policies more when discount factor allows reward to propagate relatively far enough to cause discrepancy but not to all states.

6 Q-LEARNING

For the Q-Learning algorithm, we track convergence using mean temporal difference error (delta between Q-values) over a series of 1000 iterations, as well as Mean V. Similarly to other sections, we analyse policies and Given that

Q-Learning requires more tuning and experiment, we decided to structure this section a bit differently and discussing some key observations found in our journey.

6.1 Forest Management



(a) Mean TD Error vs Iterations (1st run)

(b) Mean V vs. Iterations (1st run)

(c) Resulting QL Policy with $\gamma = 0.99$ (1st run)

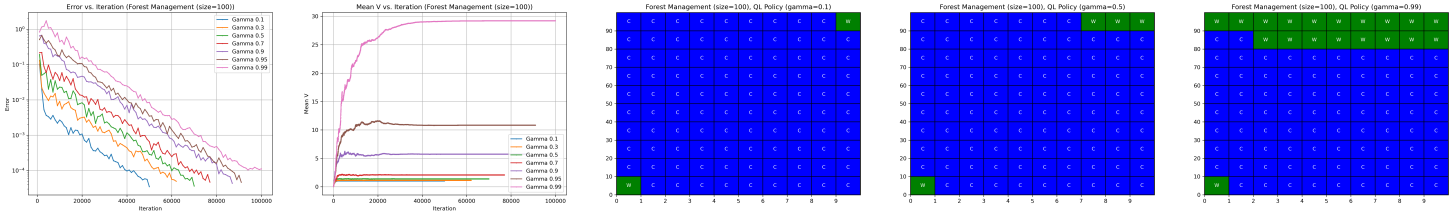
(d) Analysis: TD error by ϵ with $\gamma = 0.99$

(e) Analysis: Mean V by ϵ with $\gamma = 0.99$

Figure 8—Forest Management: 1st Q-Learning run and following analysis

The summary of running Q-Learning on the Forest Management problem the first time and following analysis is depicted in Figure 8. As one can notice, after the first try of Q-Learning algorithm, it converged in terms of both TD Error (decreasing, has fallen below 10^{-4} with decaying learning rate α with the constant number of iterations) depicted in Figure 8a and Mean V shown in Figure 8b. However, there were two things that made us question the goodness of the achieved policy: (i) values of Mean V algorithm converged to are much smaller than in VI/PI and (ii) the resulting policies were far from optimal theoretical and derived by PI/VI algorithms. While (i) can be explained by a specific choice of learning rate α and its decay factor, (ii) was raising concerns.

Following this, the first thing we tried to analyse were the values of epsilon (rate of times Q-Learning chooses random action instead of optimal) with the rest of the parameters being fixed, given that in the 1st run this parameter was setup arbitrarily to 0.1. You can see the results of this analysis in Figure 8a and Figure 8e. While all values of epsilon demonstrate the same convergence in terms of the mean Temporal Difference error, they converged to very different values of Mean V. This suggested that the choice of the epsilon was important to improve resulting policy (i.e., higher exploration might lead to policies with higher reward).



(a) Mean TD Error vs Iterations

(b) Mean V vs. Iterations

(c) Resulting QL Policy with $\gamma = 0.1$

(d) Resulting QL Policy with $\gamma = 0.5$

(e) Resulting QL Policy with $\gamma = 0.99$

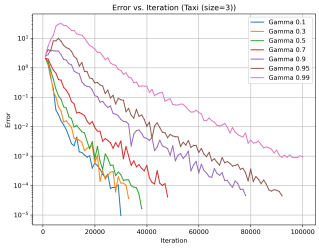
Figure 9—Forest Management: Q-Learning final run

Using insights from this analysis, we were able to improve the policies (increase number of correct actions according to PI/VI optimal policies), but there was definitely a big room for improvement. Since epsilon was one of the parameters controlling exploration-exploitation trade-off, we decided to continue digging in this direction. This brought us to the parameter controlling the time when the episode is considered finished and algorithm performs a random restart. We believed that this might have been affecting the agent and it did not visit majority of the states where it was preferring Wait actions. It turned out to be exactly the case. In the 1st run, we set up random restart every time algorithm performed Wait action and after that transition to the state $s = 0$ (fire occurred). This was a mistake to do, since the agent could become stuck for a long time in one episode once it reached the state $s = 0$ after a Cut action. Changing the condition of the random restart to *"restart whenever transition to the youngest state happens"* helped to solve the issue and achieved optimal policies. Please refer to Figure 9, which completely summarises the final run.

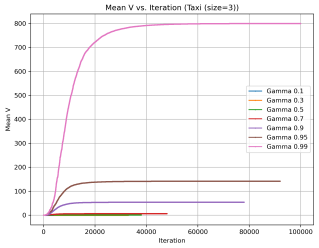
6.2 Taxi

Before running Q-Learning, on the Taxi problem in the grid world 10×10 (2000 states), we decided to conduct a simpler experiment on the problem of smaller size (3×3 grid, 180 states). The results for this run are presented in Figure 10.

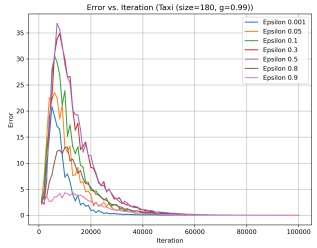
We can see from Figure 10a and Figure 10b that the algorithm converges in terms of temporal difference and Mean V for the Taxi MDP with all discount factors. To get an understanding whether the agent learns anything useful we ran the resulting policy in $N = 31$ random episodes and compared with results obtained from VI/PI policies for discount rates 0.9, 0.95 and 0.99. As a result, QL-based policy matched both VI and PI policies in terms of accumulated reward completely. We additionally checked the importance of the exploration-exploitation in this problem to get insights for a bigger one (Figure 10c and Figure 10d). We learnt that in Taxi problem, higher rate of exploration ϵ would lead to convergence to a lower value of Mean V. We believe that this suggestion comes from the fact that the problem has 6 available action and thus random choice instead of optimal heavily affects convergence. Based on this we will try to start with low epsilon for a bigger space as well, since a higher epsilon there could lead to an even worse problem.



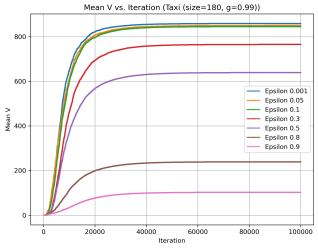
(a) Mean TD Error vs Iterations



(b) Mean V vs. Iterations



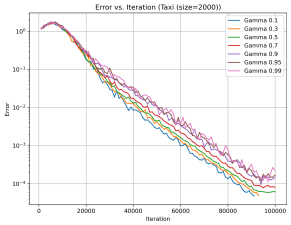
(c) Analysis: TD Error by ϵ with $\gamma = 0.99$



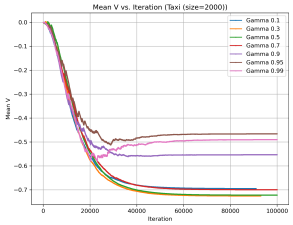
(d) Analysis: Mean V by ϵ with $\gamma = 0.99$

Figure 10—Taxi: Q-Learning run on 3x3 world (180 states) and analysis

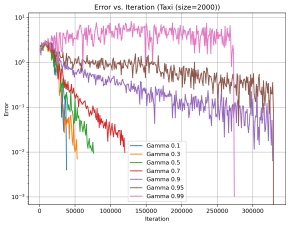
In our 1st run, in which we used setup analogus to the one we utilised for 3×3 grid world, Q-Learning was able to make a meaningful progress in terms of TD error (Figure 11a) almost reaching a threshold value of $5 \cdot 10^{-5}$. However, for all discount factor Mean V was falling throughout iterations. This, as was tested, resulted in a policy that was not able to complete any of the generated taxi episodes and deliver passengers. In order to fix this, we tried to analyse which parameters tuning could imrpove the convergence and produced policy. We excluded ϵ parameter as a an important contributing factor by looking at the spread of the expected rewards given different values of epsilon first. We then tried to increase the length of the episode (steps before the restart) from 10 to 100, which slightly improved the learnt value function. We then decided to drop decay factor from the learning rate, risking not having TD convergence. We set it to its maximum value instead, without proceeding to the decay. This way, we were trying to ensure that changes first steps are not overinflated compared to the last steps, since agent should perform better with time, but we put it in a very difficult environment where we do not value later learning as much. This helped to move the convergence faster and, even more, produced better policies. The next step we took was increasing the number of iterations 5 times, since previously set number of iterations was not enough for algorithm to converge in terms of the Mean V and TD error. The algorithm still had room for improvement in terms of Mean V, but showed good signs of convergence wrt Temporal Difference and even reached early termination threshold. We present final convergence charts in Figure 11c and Figure 11d. The obtained policies were then evaluated against optimal VI and PI policies: it reached the same reward as VI/PI in 70%, 65% and 71% of the cases for discount factors of 0.9, 0.95 and 0.99 respectively, which might be considered as a success for a model-free algorithm on such a big problem.



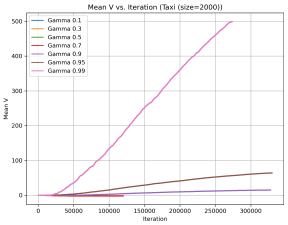
(a) 1st run: Mean TD Error vs Iterations



(b) 1st run: Mean V vs. Iterations



(c) Final run: Mean TD Error vs Iterations



(d) Final run: Mean V vs. Iterations

Figure 11—Taxi: Q-Learning run on 10x10 world (2000 states) and analysis