I. Assignment 4 Markov Decision Processes

II. Introduction

In this assignment, we delve into the practical implementation of various algorithms within the realm of Markov Decision Processes (MDPs). Our exploration is structured around two distinct MDP scenarios: a small-scale grid problem and a large-scale non-grid problem. This selection is intentional, designed to showcase the strengths and weaknesses of each algorithm in varying contexts.

We will be employing three algorithms to tackle these MDPs, focusing on their performance across different scales of state space. A key aspect of our investigation is the progressive increase in state space size for both MDPs. This approach allows us to observe how each algorithm adapts to changes in scale, providing valuable insights into their efficiency, scalability, and robustness under different conditions.

The objective is to gain a comprehensive understanding of how these algorithms operate in diverse MDP environments, emphasizing the impact of state space size on their performance. Through this assignment, we aim to not only grasp the theoretical underpinnings of MDPs but also acquire practical skills in implementing and evaluating MDP algorithms in varied situations.

III. Hypothesis

Differential Performance in Algorithms: Policy Iteration (PI), Value Iteration (VI), and Q-Learning will exhibit distinct performance profiles when applied to the small grid and large non-grid MDP problems. It is hypothesized that VI and PI will differ in their convergence speeds, with one potentially being more efficient than the other in terms of iteration count and computational resources. Q-Learning, being a model-free method, is expected to behave differently, especially in terms of convergence criteria and time.

Impact of State Space Size on Algorithm Efficiency: The efficiency of PI, VI, and Q-Learning will be significantly affected by the size of the state space

Convergence to Optimal Policies: Despite differences in their operational mechanisms and efficiency, it is expected that PI, VI, and Q-Learning will ultimately converge to similar optimal policies for each MDP, provided that sufficient iterations are allowed

Scalability and Complexity Management: As the state space increases, the scalability of PI, VI, and Q-Learning will be tested. It is anticipated that VI might show better scalability in larger state spaces due to its straightforward computation mechanism, while PI might struggle with larger state spaces due to its iterative policy evaluation and improvement steps. Q-Learning's performance in scalability is less predictable and could vary based on its parameter settings and the effectiveness of its exploration strategy.

IV. Frozen Lake

The Frozen Lake problem, set in an 8x8 grid-world environment within the context of Markov Decision Processes (MDPs), is exemplified using the gymnasium library to introduce significant stochastic elements. Each grid cell, with a high 90% probability of being a frozen tile, poses a reasonable challenge. This is further compounded by the 'is_slippery=True' condition, which introduces uncertainty in movement: the agent has only a one-third chance of moving in the intended direction, and a two-thirds probability of sliding in one of the perpendicular directions. This setup dramatically elevates the complexity, as the agent must navigate to a designated goal state while contending with the high risk of falling into holes and the unpredictability of each action. Such a configuration demands sophisticated strategy and parameter tuning from learning algorithms like Q-learning or Policy Iteration, highlighting the need for effective balance between exploration and exploitation, and adaptation to high environmental stochasticity.

V. Forest Management

The Forest Management Problem, a quintessential example in Markov Decision Processes (MDPs), aptly demonstrates decision-making under uncertainty in a dynamic environment. In this setup, a forest is managed through two actions: 'Wait' and 'Cut', decided annually. The primary goals are to preserve an old forest for wildlife conservation and to generate revenue through timber sales. However, this is complicated by the probability (p = 0.1) of a forest fire each year, adding a layer of unpredictability to the management strategy. The challenge lies in devising a policy that maximizes cumulative rewards over time, balancing the immediate benefits against long-term consequences. This reflects the essence of MDPs in sequential decision-making, where strategies must account for both immediate and future rewards, emphasizing sustainability and adaptability. For this specific assignment, we're using mdptoolbox.example.forest with 300 states and the two aforementioned actions. In this model, executing the 'Wait' action in the forest's oldest state yields a reward of 4, while choosing to 'Cut' in the same state offers a reward of 2. The constant fire probability of 0.1 further adds to the complexity, making the task of finding the optimal policy both challenging and illustrative of real-world decision-making dilemmas in uncertain and dynamic environments.

VI. Value Iteration

Value Iteration (VI) is a dynamic programming algorithm widely used in reinforcement learning and Markov Decision Processes (MDPs) for computing the optimal state value function and policy. It starts with an arbitrary value function and iteratively updates this function by evaluating all possible actions and selecting the one that maximizes value according to the Bellman optimality equation. Unlike Policy Iteration, which alternates between policy evaluation and improvement, VI updates the value function in a single, more streamlined step. This process ensures convergence to the optimal values without the need for separate policy evaluation. As a result, VI is recognized for its direct and efficient computation of

optimal policies, particularly suitable for environments where transition probabilities and rewards are clearly defined.

In this assignment, we are utilizing the hiive.mdptoolbox.mdp.ValueIteration module. The convergence criterion for VI in this context is based on the stabilization of the value function: the algorithm is deemed to have converged when the maximum change in the value function across all states between successive iterations is less than a predefined threshold. For our purposes, we are adhering to the default threshold value of 0.01, a standard setting that balances computational efficiency with the precision of the resulting policy. This threshold is a critical parameter in ensuring that VI achieves an optimal balance between performance and computational resource utilization.

VII. Policy Iteration

Policy Iteration (PI) is an established algorithm in the realm of Markov Decision Processes (MDPs) for identifying optimal policies. The algorithm initiates with a randomly selected policy and advances through two main iterative phases. In the first phase, it conducts policy evaluation, calculating the expected returns from each state. This calculation integrates immediate rewards, the discounting of future rewards, and the probabilities of state transitions. The state value function is updated repeatedly during this phase until it reaches convergence, typically using dynamic programming techniques. The second phase involves policy improvement, where actions are chosen to maximize expected returns based on a one-step look-ahead strategy. The algorithm alternates between these evaluation and improvement phases until the policy reaches a point of stability, indicating convergence to both an optimal policy and its corresponding value function. Policy Iteration is especially effective in reinforcement learning contexts where the environmental model is known, as it systematically combines policy evaluation with refinement.

For determining convergence in Policy Iteration, the key criterion is the consistency of the policy over successive iterations. When the policy remains unchanged following a cycle of evaluation and improvement, it signifies that convergence has been achieved. This stability criterion ensures that the algorithm has arrived at an optimal policy.

In this assignment, we are employing hiive.mdptoolbox.mdp.PolicyIteration. This implementation adheres to the standard approach of Policy Iteration, where the convergence is determined by the stability of the policy across iterations, ensuring the identification of an optimal policy in the given MDP scenario.

VIII. Q-Learning

Q-Learning, a groundbreaking algorithm in the field of reinforcement learning, is renowned for its guaranteed convergence to the optimal policy, distinguishing it from many earlier methodologies. Introduced by Watkins in 1989, the algorithm's convergence properties were further clarified and detailed in a study by Watkins and Dayan in 1992. This solid theoretical foundation contributes to Q-Learning's robustness

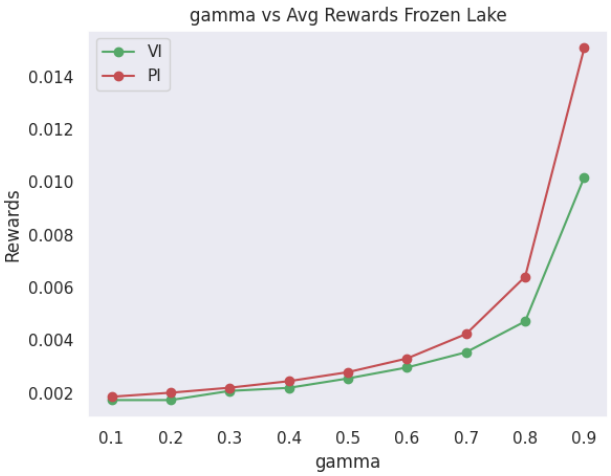

Fig. 1. gamma vs Avg Rewards Frozen Lake VI PI

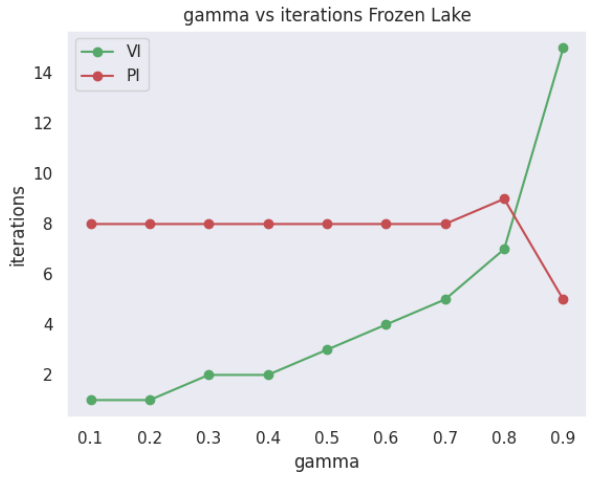

Fig. 2. gamma vs iterations Frozen Lake VI PI

and adaptability, making it highly effective across a range of reinforcement learning scenarios, especially in situations where environmental models are either unknown or too intricate to utilize directly.

In applying Q-Learning, particularly with the use of the hiive.mdptoolbox.mdp.QLearning module, there are several critical parameters that need careful calibration to optimize the algorithm's performance like learning rate (alpha), the exploration rate (epsilon), and the discount factor (gamma)

IX. Analysis

A. *comparing different values of gamma*

Fig 1 and Fig 4 showing comparison of reward with different value of gamma. we can see reward steadily increases with gamma and after 0.8 it increases significantly. This is because as gamma increases, the algorithm starts to consider future rewards more significantly. This often leads to strategies that may yield lower immediate rewards but higher long-term

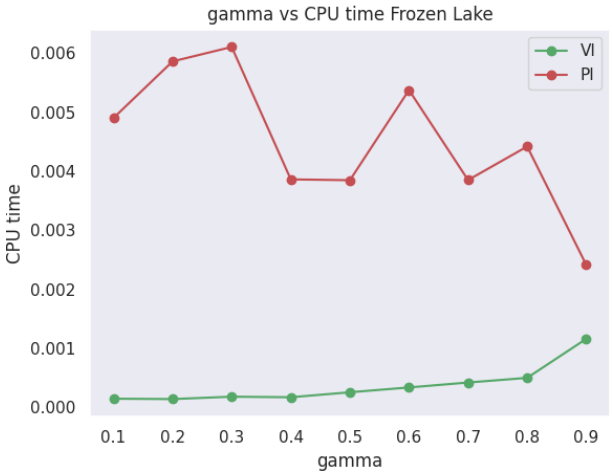

Fig. 3. gamma vs CPU time Frozen Lake VI PI

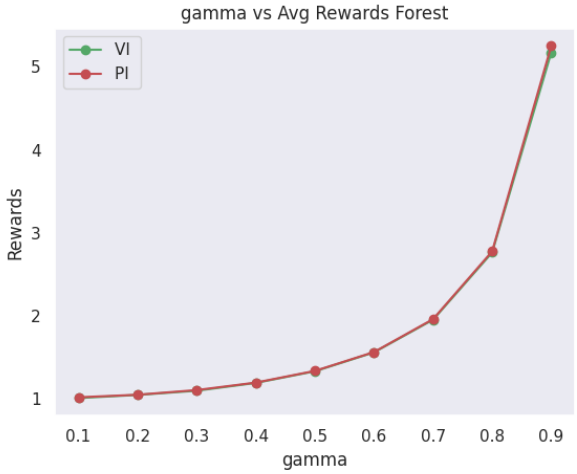

Fig. 4. gamma vs Avg Rewards Forest VI PI

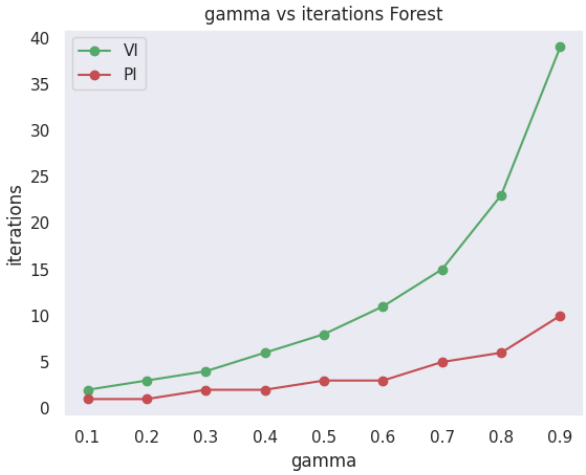

Fig. 5. gamma vs iterations Forest VI PI

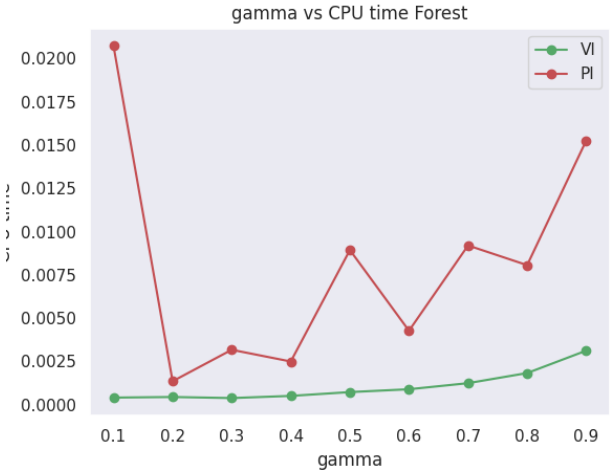

Fig. 6. gamma vs CPU time Forest VI PI

returns.from 0.1 to 0.7 range, there is a gradual shift from prioritizing immediate rewards to considering future rewards. However, the increase in total rewards is moderated by the fact that future rewards are still significantly discounted. After 0.7 the algorithms place a much stronger emphasis on future rewards. Fig1 also showing that PI gained more reward at gamma 0.9 as compare to PI. PI directly optimizes a policy, which might allow it to more effectively align with high long-term reward strategies,VI, on the other hand, incrementally improves the value function thereby deducing the optimal policy.

Fig 2 and Fig 5 showing gamma vs iterations. For Frozen Lake we can see PI iterations remains the same but VI iterations incremental increases and after gamma 0.8 shows significantly increase. This is because with a low gamma, the algorithm is short-sighted, focusing more on immediate rewards. This generally leads to fewer iterations because the algorithm doesn't have to look far into the future.As gamma increases, the value function takes longer to converge because the algorithm must account for the effects of decisions made now on the far future. Therefore, the number of iterations required to reach convergence generally increases with a higher gamma. VI is also showing similar behavior in Forest problem.

In case of PI in Frozen lake problem number of iterations remains the same and then drop and also for forest problem remains the same but slightly improve at higher gamma.In both problems, an increase in gamma shifts the focus towards considering long-term outcomes. However, the way this shift impacts policy complexity and convergence can vary based on the specific problem structure.The Frozen Lake problem has a more immediate risk-reward balance (avoiding holes to reach a goal), which may lead to quicker policy stabilization at higher gamma values. In contrast, the Forest Management problem involves a more nuanced long-term risk-reward analysis (growth vs. fire risk), which can lead to a increase in

TABLE I
FROZEN LAKE SIZE COMPARISON

	Size	Max reward	max iterations	mean cpu time
VI	8x8	0.0101873	15	0.0003232
VI	16x16	0.010864	22	0.000697
PI	8x8	0.015089	9	0.0071739
PI	16x16	0.013692	1000	0.318638

TABLE II
FOREST SPACE SIZE COMPARISON

	Size	Max reward	max iterations	mean cpu time
VI	300	5.175411	39	0.001131
VI	600	5.06266	39	0.001896
VI	3000	4.9724589	39	0.071817
PI	300	5.25329	10	0.008197
PI	600	5.140460	10	0.027119
PI	3000	5.05019	10	0.824096

iterations.

Fig 3 and Fig 6 are showing CPU time for both problems and both algorithems. VI CPU time almost remains constant from 0.1 to 0.8 and then slightly increase. which is consistence with number of rewards gained and number of iterations for higher value of gamma.As gamma increases, the value function takes longer to converge because the algorithm must account for the effects of decisions made now on the far future. In case of PI for frozen lake cpu time is higher then VI. This is consistent with higher reward and higher number of iterations we already discuss. Another factor maybe stochastic nature of Frozen Lake. PI calculates the state-value function to a certain level of precision, can be particularly resource-intensive as compare to VI which inherently focuses on the optimal action without the need to evaluate a complete policy at each step.

B. increase space size

After increasing space size of frozen lake from 8x8 to 16x16 and forest from 300 to 600 and to 3000 we can see results in Table 1 and Table 2

From table 1 we can see PI need significantly large number of iterations to converge.In PI, the policy evaluation step computes the value of each state under the current policy. With more states, this step becomes significantly more computationally intensive, requiring more iterations to converge to the true value of the current policy.Also the Frozen Lake problem is stochastic, meaning that the outcome of actions is uncertain. This uncertainty, compounded with a larger state space, makes it more challenging for the algorithm to find and confirm the optimal policy, thus requiring more iterations.

From Table 2 we can see almost no change in reward and iterations and small increase in mean cpu time. This observation suggests that the computational complexity of both VI and PI, in terms of iterations to convergence, is not heavily dependent on the size of the state space for this specific problem. This can indicate that the algorithms are quite efficient or that the problem's structure does not exponentially increase in complexity with more states.

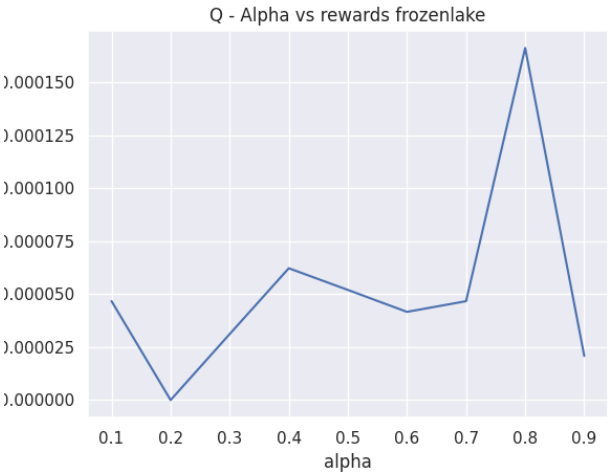

Fig. 7. Alpha vs rewards frozen lake Q Learning

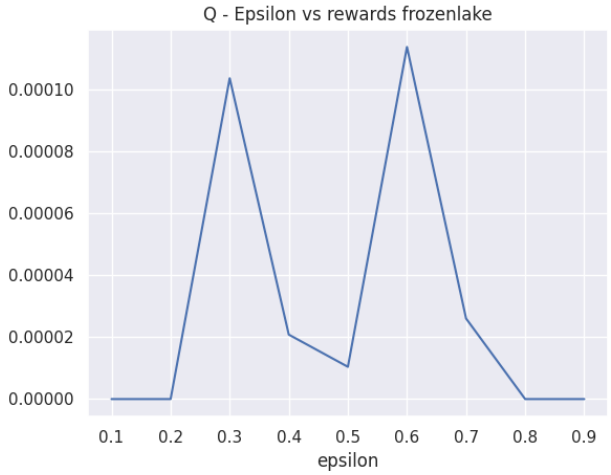

Fig. 8. Epsilon vs rewards frozen lake Q Learning

X. Q LEARNING

Fig 7 and fig 9 showing alpha vs rewards.The learning rate alpha in Q-learning determines how new information influences the existing value estimates. An alpha of 0.8 suggests a significant weight is being given to new information, which might be particularly effective in the Frozen Lake environment for quickly learning beneficial actions.In case of forest problem alpha and reward are showing almost a liner relation suggesting that it is a easy problem to learn for the agent.

Fig 8 and fig 10 showing epsilon vs rewards epsilon controls

TABLE III
FROZEN LAKE SIZE COMPARISON

	Size	Max reward	mean cpu time
Q	8x8	0.0042461	3.386911
Q	16x16	1.426512	6.07515

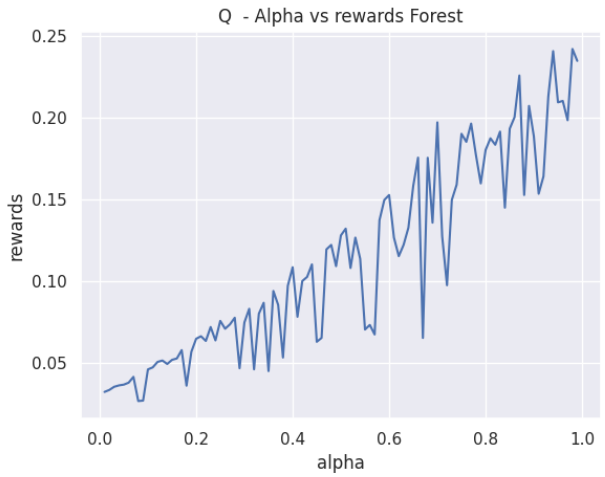

Fig. 9. Epsilon vs rewards frozen lake Q Learning

Fig. 10. Epsilon vs rewards forest

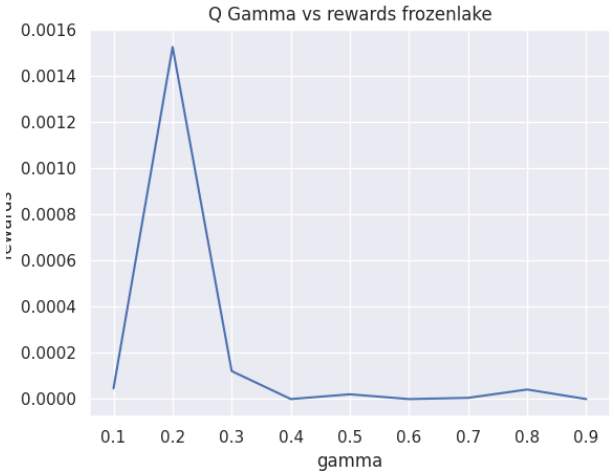

Fig. 11. Q Gamma vs rewards Forzen lake

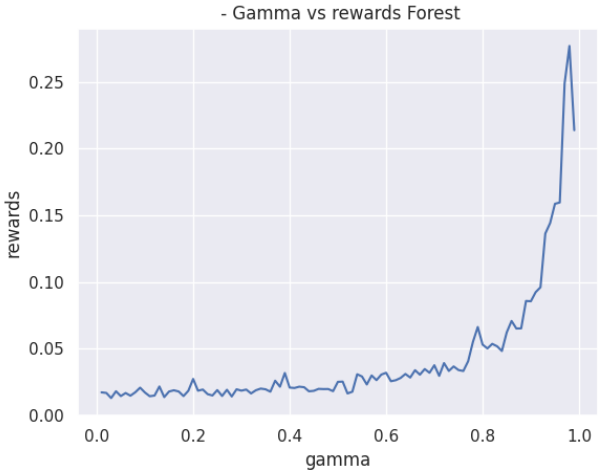

Fig. 12. Q Gamma vs rewards forest

TABLE IV
FOREST SPACE SIZE COMPARISON

	Size	Max reward	mean cpu time
Q	300	0.43987	4.24065
Q	600	0.205392	4.81683
Q	3000	0.040071	13.72717

the balance between exploration and exploitation.In case of frozenlake the first peak at 0.3 might represent an optimal balance early in the learning process, where some exploration is necessary to gather sufficient information about the environment. The second peak at 0.6 could correspond to a later stage, where increased exploration is beneficial to refine the strategy and adapt to the more nuanced aspects of the environment. In case of forest problem zig zag and horizental line suggest that randomness in the Forest Management problem interact with the exploration strategy in unpredictable ways, leading to the observed zig-zag pattern in rewards.

Fig 11 and 12 showing gamma vs rewards.For gamma vs reward in case of frozen lake spike at 0.2 gamma could be that frozen Lake problem might have an optimal balance of immediate and future rewards that is uniquely captured at that value. In forest is showing same trend as we saw in VI and PI. This is because as gamma increases, the algorithm starts to consider future rewards more significantly. This often leads to strategies that may yield higher long-term returns.

Table III showing max reward and mean cpu time for two sizes of frozen lake. we can see mean CPU time for q learning double as size of problem double. Each iteration in Q-Learning involves updating Q-values for encountered state-action pairs. More states lead to more potential updates per iteration, contributing to the increased overall time.

Table IV showing max reward adn mean CPU time for forest. we can see that mean cpu time significantly increase between space size of 600 and 3000 but show little change for size between 300 to 600. This mainly due to the exponential growth in the number of state-action pairs, increased

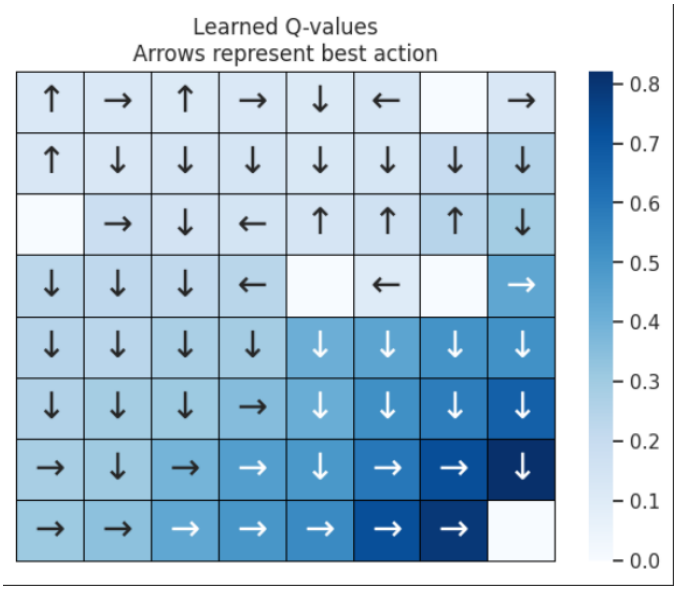

Fig. 13. Enter Caption

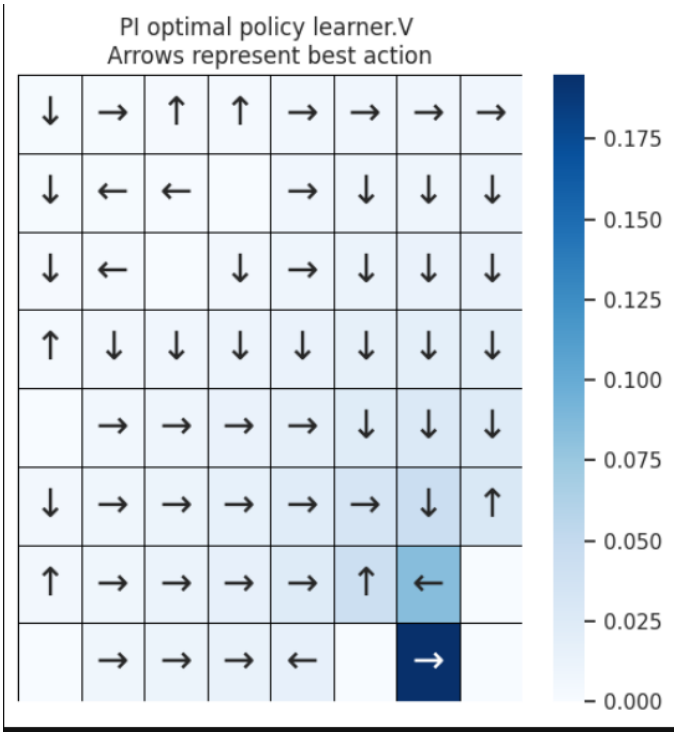

Fig. 14. Enter Caption

complexity of updates, longer convergence times, and memory constraints, which become more significant as the state space size increases.

XI. Frozen lake optimal policy output

Fig 13 to 14 showing heat map of optimal policy of VI PI and qlearning. Agent start at top left and reward is at bottom right side. Arrows showing best action and empty cell

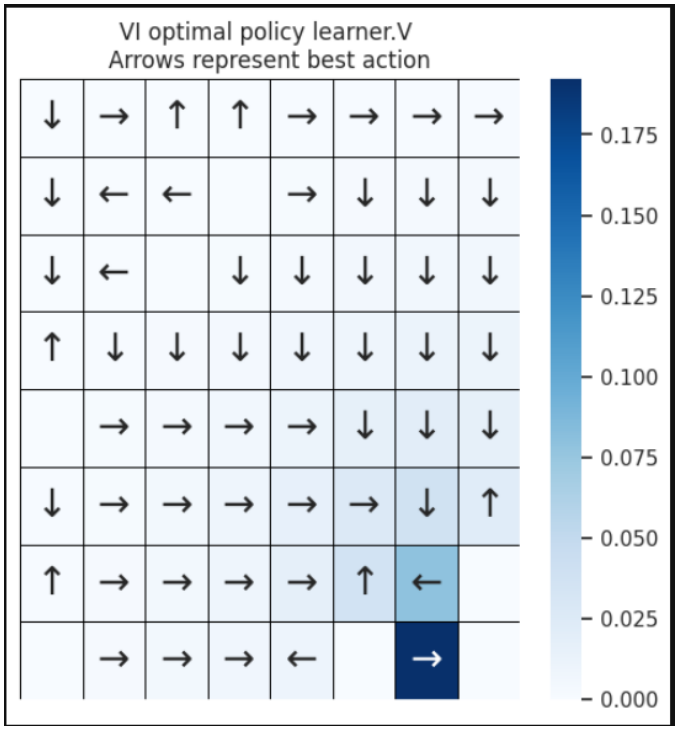

Fig. 15. Enter Caption

showing hole in ice. Darker color near reward suggest higher-valued states and illustrate the influence of reward proximity on the value function and policy. They visually convey how the algorithms guide the agent towards rewarding states, reflecting the effectiveness of the learning process and the impact of factors like the discount factor

Identical heat maps from VI and PI indicate successful convergence to the same optimal policy, validate the correctness of the algorithms, and provide insights into the problem's nature and the robustness of the solution. As we already discuss above these two algorithm show differences in CPU time and number of iterations to reach optimal solution.

XII. Conclusion

In conclusion, this report has provided a comprehensive analysis of three key algorithms - Value Iteration (VI), Policy Iteration (PI), and Q-Learning - applied to solve Markov Decision Process (MDP) problems. Through a series of comparative evaluations, we have gained insightful observations about their performance, convergence properties, and computational efficiencies. Our analysis revealed that both VI and PI consistently converge to the same solutions across various MDP problems. This consistency affirms the reliability of these algorithms in deriving optimal policies. We observed notable differences in CPU time and iterations between VI and PI, contingent on the specific nature and size of the problem. While PI generally takes fewer iterations than VI, it demands more computational time per iteration due to its comprehensive policy evaluation step. A key observation was the significant change in PI's behavior with an increase in the size of the

Frozen Lake problem. Contrary to its typical performance in smaller problems, PI required substantially more iterations to converge when the problem size was doubled, suggesting a heightened sensitivity to state space expansion. In comparison to VI and PI, Q-Learning exhibited a higher demand for iterations and CPU time. As a model-free algorithm that learns through environmental interactions, Q-Learning's extensive iteration requirement highlights its adaptability but also points to its computational intensity, especially in complex scenarios. These findings have crucial implications for the selection of algorithms in MDP contexts. VI and PI, with their model-based approach, offer efficiency and reliability in optimal policy derivation, yet their performance is influenced by the problem's scale. Q-Learning, with its inherent flexibility, stands out in scenarios where model-based methods are less feasible, albeit with greater computational demands. If we look at our hypothesis we are correct in assuming that PI and VI behave distinctly in term of convergence speed, number of iterations and computational resources. The efficiency of PI, VI, and Q-Learning was significantly affected by the size of the state space Convergence to Optimal Policies specially PI and VI converge to same policy for frozen lake. Solubility and Complexity Management: As the state space increases, the solubility of PI, VI, and Q-Learning was tested. VI shows some solubility but PI shows the opposite

REFERENCES

Other then course lecture and book

- https://www.kaggle.com/
- https://towardsdatascience.com/
- https://medium.com/
- https://datascience.stackexchange.com/
- https://chat.openai.com/