Markov Decision Processes Assignment

Chase Brooks

dbrooks43@gatech.edu

Abstract**—** This paper describes my approach to solving two different Markov Decision Process (MDP) Problems by Value Iteration, Policy Iteration, and QLearning. First, I solve the Forest Management Problem (Non-Grid World) and Frozen Lakes Problem (Grid World) via a model-based approach with Value and Policy Iteration and compare results. Then, I solve these two problems again with a model-free approach, QLearning, and compare.

1 PROBLEM OVERVIEW

1.1 Forest Management Problem

Forest management is a planning problem whereby a forest ranger has a forest full of S trees. For each tree, s, the manager has two choices: wait (0) or cut (1).
This problem is particularly interesting because we must weigh two objectives and rewards. Waiting allows wildlife to flourish and has an associated reward. Cutting makes the manager money and has a separate reward. We must create a policy that maximizes the rewards of both maintaining as old a forest as possible for the wildlife as well as making as much money as possible. Each year, there is a probability p that the entire forest burns down and forest age resets at 0. This mechanism means that while our policy is deterministic, the environment itself is stochastic. Thus, to create the policy that maximizes rewards, we must consider multiple objectives and navigate an inherently stochastic environment.

1.2 Frozen Lakes Problem

The Frozen Lakes Problem consists of a 4x4 grid-world whereby each cell corresponds to 1 of 4 possible values: Start (S), Frozen (F), Hole (H), and Goal (G). The goal of this problem is to determine the shortest path from an arbitrary starting point to the goal cell. The path must only consist of frozen cells, and entirely avoid cells with holes. Possible agent moves are up, down, right, and left. I find this problem particularly interesting because it provides a simplified version of an optimal policy search problem that can be made either deterministic or stochastic in nature. With only 16 possible states in a 4x4 world, this problem also will not be prohibitively computationally expensive to try various algorithms.
This problem is particularly interesting because moves are stochastic. This means that an agent will execute our policy's chosen move with probability p, but will choose a move not specified by our policy with probability 1-p.

2 ALGORITHMS

I first begin by comparing performance of the model-based approaches; Value Iteration and Policy Iteration.

2.1 Model-Based Learning Algorithms

Model-based learning algorithms rely on prior knowledge of a problem to function. They operate by defining transition probability and reward matrices and iteratively updating them to solve a problem. I will evaluate the performance of two forms of model based algorithms.

2.1.1 *Policy Iteration*

Policy Iteration works by first starting with an arbitrary policy (π) followed by iterative evaluations and improvements until we converge on an optimal policy. Each evaluation looks at future states, and applies a discount factor (γ) to model the tradeoff of short term and long term payoffs. The lower the discount factor, the more the policy weighs short term gains and vice versa. In finite search spaces, there are a finite number of potential policies, thus an optimal policy is guaranteed (Tokuç, October, 2021).

2.1.2 *Value Iteration*

Value Iteration works by computing a value function (v(s)) and iteratively updating until convergence. At each step, we calculate the value of each potential action and select the one that maximizes our value function. Value iteration updates the entire state value function in a single step instead of evaluating and improving iteratively like the policy iteration algorithm. Value iteration is also guaranteed to converge and the optimal policy.

2.2 Model-Free Learning

Model Free learning algorithms allow us to create policies in spaces where we do not necessarily have prior knowledge. Thus, this class of algorithms can “learn” transition probability and reward matrices as they progress through the problem. I will evaluate a QLearning algorithm and compare its performance with Value and Policy Iteration.

2.2.1 *QLearning*

QLearning algorithms do not know the transition probability and reward matrices from the outset. Instead, they work by discovering rewards as they move to new states. They empirically determine transition probability matrices by landing in a particular state and evaluating options from that state. They can also learn stochastic transitions by observing how frequently transitions between particular states occur.

2.3 Comparison of Algorithms

Model-based algorithms excel in small state spaces where we possess prior knowledge about the problem. Policy Iteration requires fewer iterations than value iteration to converge. Model-free algorithms such as QLearning typically excel in problems with no prior knowledge. Model-free algorithms are not guaranteed to converge on an optimal policy, and the empirical nature of testing various actions typically leads to longer training times than model-based algorithms.Thus, I expect the QLearner to perform worse than the model-based approaches on the forest management and frozen lake problem.

2.3.1 *Time Complexity of Model-based algorithms*

As we will explore in subsequent sections, the time complexity of each model-based algorithm will be a critical differentiator in algorithm performance under various scenarios. The time complexity of each iteration in value iteration is $O(|S^2||A|)$ where $|S|$ is the number of states and $|A|$ is the number of actions (Kaelbling, May 1996). The time complexity per iteration in policy iteration is $O(|S^2||A| + |S|^3)$ (Kaelbling, May 1996). Thus, in larger state spaces, the $|S|^3$ dominates per iteration time complexity of policy iteration and causes wall clock time to quickly explode in large state spaces. Section 3.1 will explore the impact of these time complexity differences across various state space sizes for the forest management problem.

3 EXPERIMENTS

I will first evaluate the performance of model-based algorithms in both small and large state spaces to compare performance and highlight strengths/weaknesses. I will then rerun these experiments with the model-free QLearning to highlight its strengths and weaknesses in large and small state spaces. For all experiments, convergence is defined as when successive iterations yield a less than < 0.01 change in the value function.

3.1 Model-based Forest Management Problem: Number of States = 1000

I will first begin analyzing performance of each algorithm over various state space sizes. Figure 1 shows that as the number of states increases, value iterations require significantly more iterations to converge on an optimal policy than policy iteration. Even though policy iteration requires fewer iterations, the wall clock time is significantly higher than value iteration in larger state spaces. Table 1 shows that as the number of states surpasses 200, the number of iterations required to reach convergence does not change. Despite a constant number of iterations to converge, the ever increasing wall clock time indicates that the computational complexity of each iteration increases as state space increases since more potential policies/value functions need to be evaluated. Thus, I will compare performance between the two algorithms with 1000 states to hopefully highlight these differences.

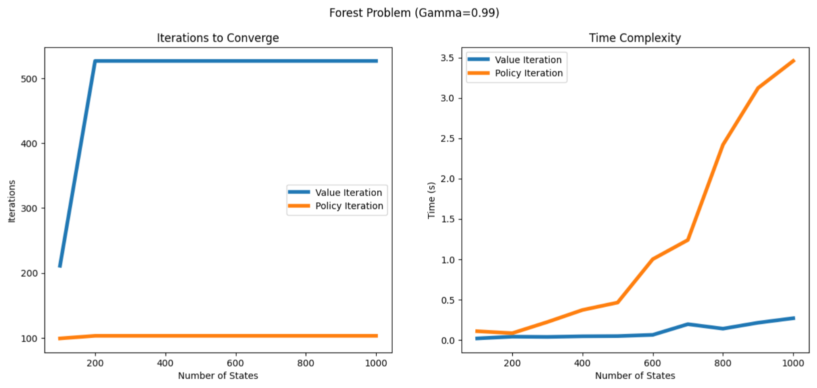

Figure 1— Forest problem iterations & time complexity of convergence (γ=0.99)

Table 1—Number of Iterations to Converge with various state space sizes for $\gamma = 0.99$

Number of Iterations to Convergence	Policy Iteration	Value Iteration
States = 100	99	211
States = 200	103	527
States = 1000	103	527

3.1.1 *Effect of Discount Rate on Convergence*

Discount factor (γ) is one key parameter that we can tune to change each algorithm's weighting of short vs long term payoffs. Table 2 shows that across several different discount factors, policy iteration converges to an optimal policy in fewer iterations that value iteration. As the discount rate increases, so too does the number of iterations required to reach an optimal policy since a larger gamma means we discount future rewards less and must consider additional long tail policy combinations. However, both algorithms do eventually converge to the optimal policy. Figure 2 shows that, although the reward value changes at various discount rates, both algorithms eventually converge on the optimal policy with the highest reward for each given discount factor.

Table 2— Number of iterations to converge on the optimal policy for a large state space forest management problem

Number of iterations to convergence	Policy iteration	Value iteration
Discount: 0.9	18	71
Discount: 0.95	34	146
Discount: 0.99	103	527

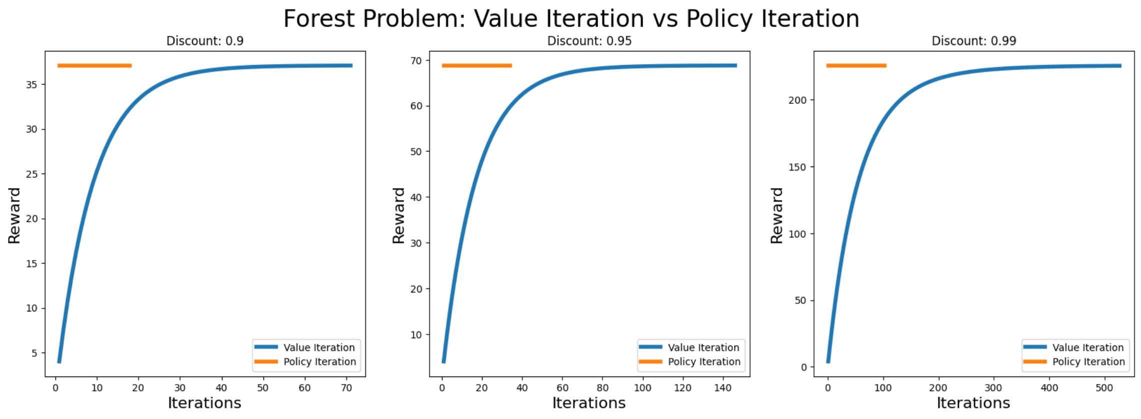

Figure 2—Number of iterations to converge on an optimal policy in a large state space forest management problem with various discount factors (γ)

3.2 Model-based 4x4 Frozen Lake Problem: Number of States = 16

Even in smaller state spaces, policy iteration requires fewer iterations to converge on the optimal policy than value iteration. Similarly to the forest management problem with a larger state space, policy iteration converges faster than value iteration, but both algorithms eventually converge on the optimal policy. Figure 3 shows that as γ increases, so too does the number of iterations to converge. Additionally, it appears that an increasing γ has a larger impact on driving up the number of iterations for value iteration than it does for policy iteration.

Table 3— Number of iterations to converge on the optimal policy for a small state space frozen lake problem

Number of Iterations to Convergence	Policy Iteration	Value Iteration
Discount: 0.85	4	16
Discount: 0.90	6	26
Discount: 0.95	5	52

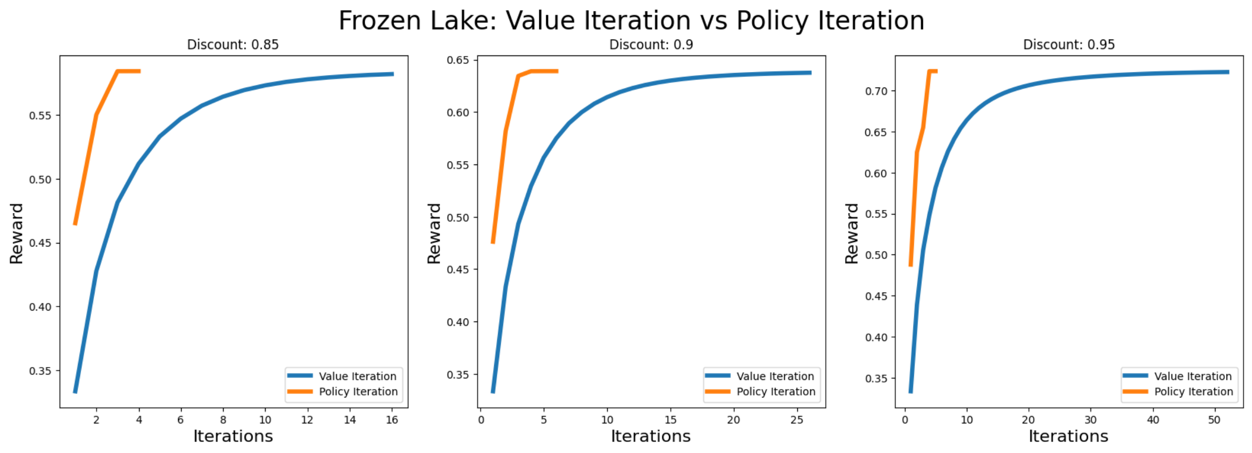

Figure 3—Number of iterations to converge on an optimal policy in a small state space frozen lake problem with various discount factors (γ)

3.3 QLearner

To explore QLearning, I will rerun both the large state space forest management problem and the small state space frozen lake problem. I will also measure performance across various hyperparameters that impact a QLearner's search for the optimal policy.

3.3.1 *Exploration Strategy: Effect of Learning Rate (α) on QLearner Convergence*

The Learning Rate (α) determines how quickly a QLearner overrides old information with new information. A lower α means that this process occurs more slowly, and a higher α indicates more rapid incorporation of newly learned information into the Q-table. In both the frozen lake and forest problems, it appears than lower alphas yield better results since a lower α results in faster convergence because the algorithm relies more heavily on past information.

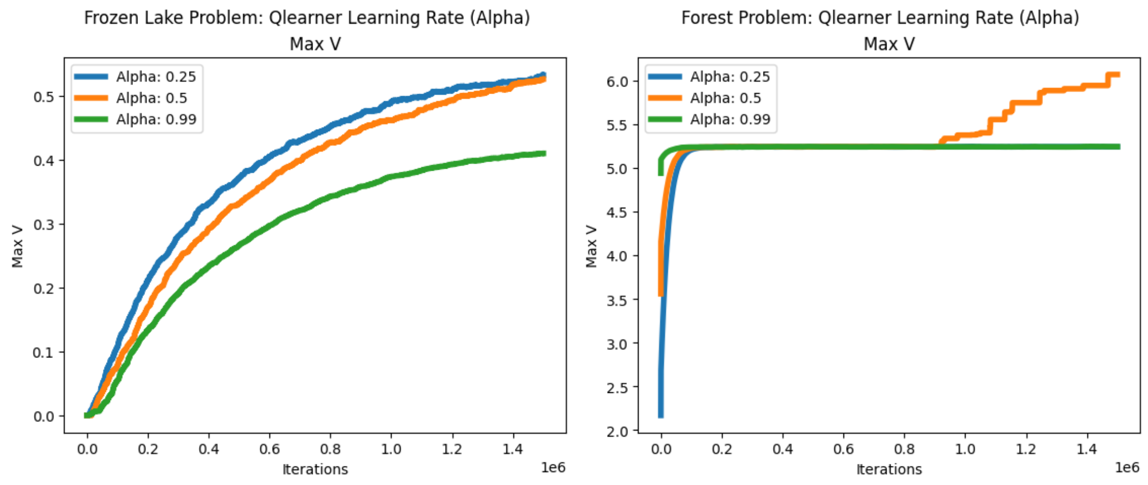

Figure 4—Effect of Learning Rate (α) on convergence of QLearner for forest and frozen lake problem

3.3.2 *Exploration Strategy: Effect Alpha Decay Rate on Convergence*

The Learning Rate (α) effectively determines how quickly the Q tables are updated which is how the QLearning algorithm "learns" its environment. Another parameter we can tune is the α Decay. The α Decay allows a QLearner to decrease its learning rate as the number of iterations increase. Effectively, this means that as a QLearner sees more and more data, it begins to trust prior data more than newly discovered data. Figure 4 shows the effect of alpha decay rate on mean/max estimated value function (V) at each iteration. In the frozen lake problem, α Decay = 0.95 appears to outperform both 0.9 and 0.99. There does not appear to be much differentiation in performance across α Decays in the forest problem.

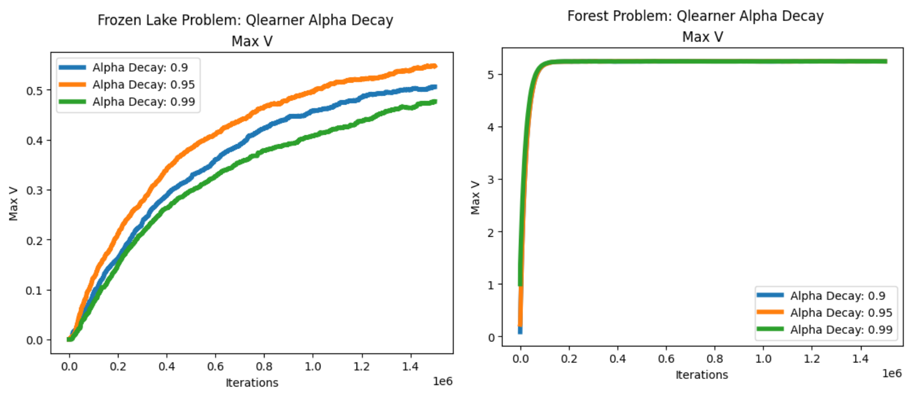

Figure 5—Effect of Learning Decay Rate (α decay) on convergence of QLearner for forest and frozen lake problem

3.3.3 *Exploration Strategy: Effect Epsilon-greedy action selection parameter on Convergence*

By altering ε, we can balance the QLearner's exploration of new policies vs. exploitation of existing policy to maximize rewards. At each iteration, the QLearner will either select a random action with probability ε, or choose the best known action according to its Q-Table with probability 1-ε. Higher ε values mean the QLearner is more likely to select random actions (ie explore) and lower ε values mean the QLearner will select best action already known (ie exploit current knowledge). For the forest problem, it appears that lower epsilon values outperform higher epsilon values, which means that our QLearner should more heavily weight exploitation than exploration as the number of iterations increases. For the forest problem, there appears to be some differentiation in performance across lower epsilon values, but once the QLearner passes 400,000 iterations, performance begins to converge across epsilons.

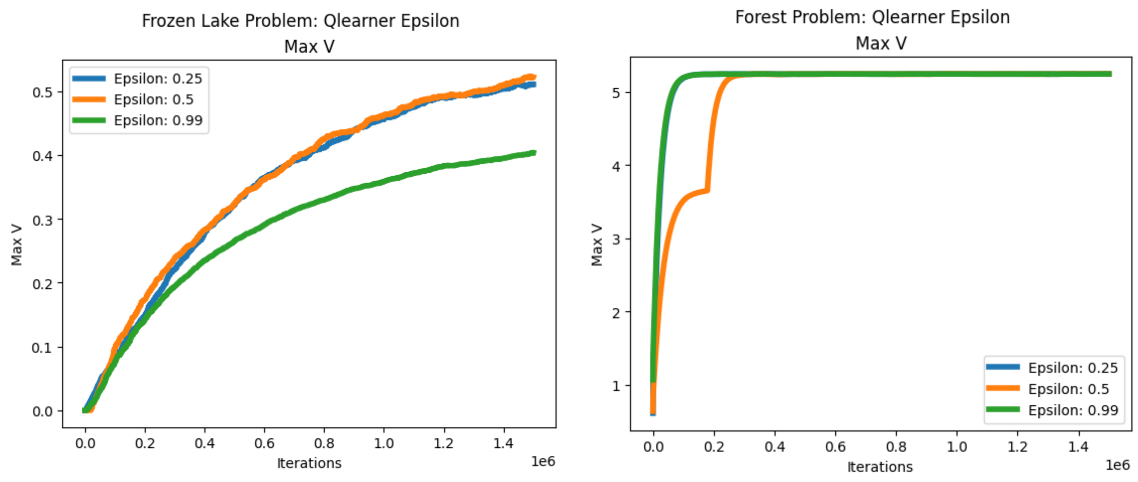

Figure 6—Effect of Epsilon on convergence of QLearner for forest and frozen lake problem

3.4 QLearner vs. Model-based Algorithms on the Forest Problem

Even after tuning various hyperparameters of the QLearner, it still performs worse across all major metrics than value and policy iteration. Table 4 compares performance between the QLearner and Value/Policy Iteration across various state space sizes. While QLearning takes significantly longer to converge on a sub-optimal policy, individual actions within its policy have a high degree of overlap with the optimal policies generated by value and policy iteration for the same state. For example, with a 500 state space, 86% of actions in the QLearner policy are exactly the same as Value and Policy Iteration policies! This is especially impressive since the QLearner, with no prior knowledge of the problem, still generates a policy that performs reasonably well in most situations.

Table 4— QLearner Performance on Forest Problem vs. Value and Policy Iteration

State Size	% of QL Policy that is optimal	QLearning Time	Value Iteration Time	Policy Iteration Time
S = 100	80%	72.3s	0.01s	0.06s
S = 200	86%	78.5s	0.01s	0.02s
S = 500	86%	82.5s	0.01s	0.10s
S = 1000	78%	94.6s	0.11s	0.77s

4 CONCLUSION

In situations where we have prior knowledge of the problem space, model-based approaches appear to outperform even a tuned QLearner. This follows intuition because if our problem space is a world where we can perfectly predict a reward for a given action, then applying model-based algorithms to plan our actions and guarantee an optimal policy is preferable. However, Table 4 shows that QLearners can still generate policies that perform reasonably well. In reality, model-free algorithms truly shine when we apply them to problems where we have no prior knowledge of the environment.

In other words, model-based algorithms take a deductive learning approach to determine the best action given its understanding of its environment. Model-free algorithms take an inductive learning approach where past experience and statistics are used as empirical evidence to determine the best action.

5 REFERENCES

1. Tokuç, A. A. (2021, October 13). *Value iteration vs. policy iteration in reinforcement learning*. Baeldung on Computer Science. Retrieved April 17, 2022, from https://www.baeldung.com/cs/ml-value-iteration-vs-policy-iteration
2. Kaelbling, L. P. (1996, May 1). Computational complexity. Retrieved April 17, 2022, from https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/node22.html