Machine Learning
Markov Decision Processes

Jorge Salvador Aguilar Moreno

I. Introduction

Markov Decision Processes (MDPs) are a mathematical framework for modeling decision-making in environments where outcomes are partly random and partly under the control of a decision-maker. An MDP is defined by a tuple (S, A, P, R, γ), where S represents the set of states, A represents the set of actions, $P(s'|s,a)$ is the transition probability from state s to s' under action a, $R(s,a)$ is the reward function, and γ is the discount factor. MDPs are foundational in Reinforcement Learning (RL), where the goal is to learn an optimal policy π^* that maximizes the expected cumulative rewards.

RL algorithms can be categorized into two types: model-based and model-free approaches. Model-based algorithms utilize the transition probabilities and reward functions of the environment, whereas model-free algorithms rely on sampling interactions to learn optimal policies without explicit knowledge of the environment dynamics. This report investigates both approaches, applying them to two MDP problems: the discrete-state Blackjack problem and the continuous-state Cart Pole problem.

A. Model-Based Algorithms

Model-based algorithms solve MDPs by leveraging the known dynamics of the environment, specifically the transition probabilities $P(s'|s,a)$ and the reward function $R(s,a)$. Two prominent model-based approaches are Value Iteration and Policy Iteration.

Value Iteration: Iterative algorithm that updates the value function $V(s)$ for each state based on the Bellman optimality equation:

$$V(s) = \max_{a \in A} \sum_{s' \in S} P(s'|s,a)\big[R(s,a) + \gamma V(s')\big].$$

The algorithm initializes $V(s)$ arbitrarily, and repeatedly applies the update rule until the values converge within a specified tolerance θ. The optimal policy $\pi^*(s)$ is derived by selecting the action that maximizes the expected value:

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s'|s,a)\big[R(s,a) + \gamma V(s')\big].$$

This approach, described in [1] and [3], guarantees convergence to the optimal value function V^* and policy π^*.

Policy Iteration: Algorithm alternates between two steps: policy evaluation and policy improvement. In the policy evaluation step, the value function for a fixed policy π is computed iteratively using the equation:

$$V^\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} P(s'|s,a)\big[R(s,a) + \gamma V^\pi(s')\big].$$

In the policy improvement step, the policy is updated to maximize the value function:

$$\pi'(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s'|s,a)\big[R(s,a) + \gamma V^\pi(s')\big].$$

The algorithm terminates when the policy stabilizes, ensuring convergence to the optimal policy π^* [3].

B. Model-Free Algorithms

Model-free algorithms do not require explicit knowledge of the environment dynamics. Instead, they learn optimal policies directly from interaction with the environment through trial and error. Two widely used model-free RL algorithms are Q-Learning and SARSA.

Q-Learning: Off-policy algorithm that learns the optimal action-value function $Q(s,a)$ by iteratively updating it based on observed transitions. The update rule is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\big],$$

where α is the learning rate, γ is the discount factor, and s' is the next state. By selecting the action a that maximizes $Q(s,a)$, the algorithm converges to the optimal policy π^* [4].

SARSA (State-Action-Reward-State-Action): On-policy algorithm that updates the action-value function $Q(s,a)$ using the action taken by the current policy. The update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big[R(s,a) + \gamma Q(s',a') - Q(s,a)\big],$$

where a' is the next action chosen by the current policy. This approach ensures the algorithm learns the value of the policy being followed, making it more stable in some environments [5].

II. "Small" MDP Problem: Blackjack

A. Problem Description

Blackjack is a card game where the objective is to beat the dealer by obtaining cards that sum closer to 21 than the dealer's cards without exceeding 21 [6]. The game environment, as implemented in Gymnasium, models a simplified version of Blackjack with discrete states and actions. The blackjack problem is particularly interesting because many gambling games have been extensively studied using game theory and ML techniques. By applying RL algorithms, we

can derive feasible and optimal strategies for playing the game effectively. However, it is important to note that, as with most gambling scenarios, the house has a statistical advantage. As the number of games approaches infinity, the expected reward for the player tends to be negative.

State Space: There are 290 discrete observable states, formed by:

- 29 player hands: H4–H21 (hard totals 4 to 21), S12–S21 (soft totals 12 to 21), and BJ (blackjack).
- 10 dealer face-up cards: 2–9, T (10), and A (ace).

Action Space: The player chooses to:

- **Stick (0):** Stop drawing cards.
- **Hit (1):** Draw an additional card.

Rules and Rewards:

- Cards: Numbered cards (2–9) are worth their face value, face cards (J, Q, K) are worth 10, and aces are worth 1 or 11 (usable ace).
- The player wins $(+1)$ if their sum is closer to 21 than the dealer's without exceeding 21. Loss results in a reward of -1, while a draw gives 0.

Game Flow:

- The player decides to hit or stick until sticking or busting (sum > 21).
- The dealer draws cards until their sum is at least 17.
- The episode ends when the player sticks, busts, or the game resolves.

B. Hypothesis

For the blackjack problem, a simple MDP with 290 discrete states, it is hypothesized that model-based algorithms, such as V.I. and P.I., will outperform model-free algorithms like Q-Learning and SARSA. The finite and fully observable state space enables model-based methods to directly compute optimal policies using known transition probabilities and rewards, ensuring faster convergence and shorter runtimes.

Another hypothesis is that Value Iteration is expected to outperform Policy Iteration due to its direct updates of value functions, whereas Policy Iteration's alternating evaluation and improvement steps require additional computation. Convergence issues are not anticipated, and model-free algorithms are likely to exhibit longer runtimes and variability in performance due to their reliance on trial-and-error learning.

C. Experimental Methodology

The objective of these experiments is to evaluate the performance, runtime, convergence behavior, and strategies of RL algorithms in the "simple" Blackjack problem.

For model-based algorithms, V.I. and P.I. are tested with a range of discount factors, $\gamma = [0, 0.05, 0.2, 0.4, 0.6, 0.8, 1.0]$, using a convergence criterion of $\theta = 1 \times 10^{-10}$. Metrics such as convergence plots, training time, and average rewards over 100,000 iterations are recorded for each algorithm.

For model-free algorithms, Q-Learning and SARSA are evaluated under the same range of discount factors ($\gamma = [0, 0.05, 0.2, 0.4, 0.6, 0.8, 1.0]$) with a convergence criterion of $\theta = 1 \times 10^{-10}$. Each algorithm is trained for 10,000 episodes. The following parameters are used for learning rate and exploration strategies:

Learning rate: $\alpha_{\text{init}} = 0.5$, $\alpha_{\text{min}} = 0.01$, $\alpha_{\text{decay}} = 0.5$.
Exploration: $\epsilon_{\text{init}} = 1.0$, $\epsilon_{\text{min}} = 0.1$, $\epsilon_{\text{decay}} = 0.9$.

Convergence plots, training time, and average rewards after 100,000 iterations are collected for both algorithms.

In addition, two supplementary experiments are conducted for model-free algorithms, keeping $\gamma = 0.6$ to explore the impact of parameter variations:

1) Exploration-Exploitation Trade-Off: This experiment investigates the effect of different exploration strategies:

Limited Exploration: $\epsilon_{\text{init}} = 0.5$, $\epsilon_{\text{min}} = 0.1$, $\epsilon_{\text{decay}} = 0.9$.
Balanced Exploration: $\epsilon_{\text{init}} = 1.0$, $\epsilon_{\text{min}} = 0.1$, $\epsilon_{\text{decay}} = 0.9$.
Aggressive Exploitation: $\epsilon_{\text{init}} = 1.0$, $\epsilon_{\text{min}} = 0.5$, $\epsilon_{\text{decay}} = 0.5$.
Aggressive Exploration: $\epsilon_{\text{init}} = 1.0$, $\epsilon_{\text{min}} = 0.01$, $\epsilon_{\text{decay}} = 0.99$.

2) Learning Strategies: This experiment evaluates the impact of different learning rate strategies:

Fast Learning: $\alpha_{\text{init}} = 0.9$, $\alpha_{\text{min}} = 0.01$, $\alpha_{\text{decay}} = 0.3$.
Balanced Learning: $\alpha_{\text{init}} = 0.5$, $\alpha_{\text{min}} = 0.01$, $\alpha_{\text{decay}} = 0.5$.
Slow Learning: $\alpha_{\text{init}} = 0.5$, $\alpha_{\text{min}} = 0.1$, $\alpha_{\text{decay}} = 0.8$.
Aggressive Learning: $\alpha_{\text{init}} = 0.9$, $\alpha_{\text{min}} = 0.5$, $\alpha_{\text{decay}} = 0.1$.

For both experiments, the impact on convergence, training time, and final policy quality is analyzed.

D. Analysis of Model-Based Algorithms

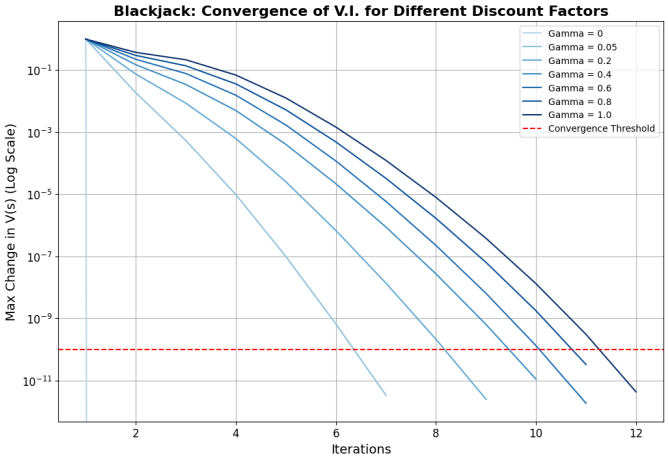

Fig. 1. Blackjack: Convergence of Value Iteration

Figure 1 illustrates the convergence of the Value Iteration (V.I.) algorithm, showing the reduction in the maximum change of $V(s)$ across iterations on a logarithmic scale. Due to the small state space (290 discrete states), convergence is achieved in no more than 12 iterations for all tested discount factors (γ).

Higher discount factors, such as $\gamma = 1.0$, take slightly longer to converge as they prioritize long-term rewards, whereas lower discount factors, like $\gamma = 0$, converge more quickly by focusing on immediate rewards, 2 iterations in this case. The red dashed line represents the convergence threshold

($\theta = 10^{-10}$), met successfully by all discount factors. This demonstrates the stability and efficiency of V.I. for a simple MDP like Blackjack.

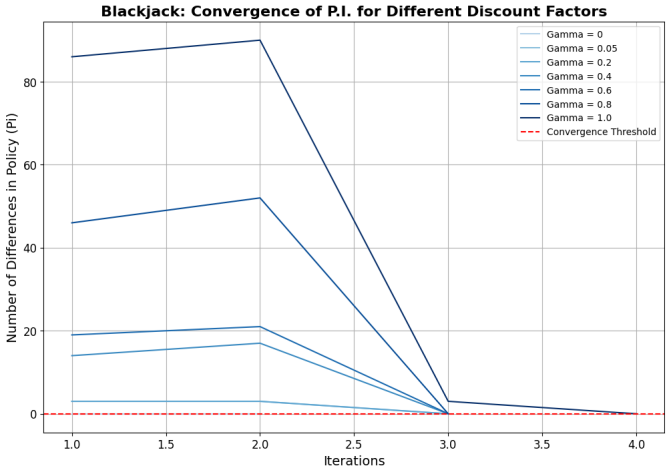

Fig. 2. Blackjack: Convergence of Policy Iteration

Figure 2 shows the convergence of Policy Iteration (P.I.) by tracking the number of differences in the policy π across iterations. Results are similar to Figure 1 because the algorithm converges quickly, in this case more than 4 iterations for all discount factors (γ), highlighting its efficiency for small state spaces. Once again, higher discount factors exhibit slightly more initial differences in the policy, reflecting a greater emphasis on long-term rewards.

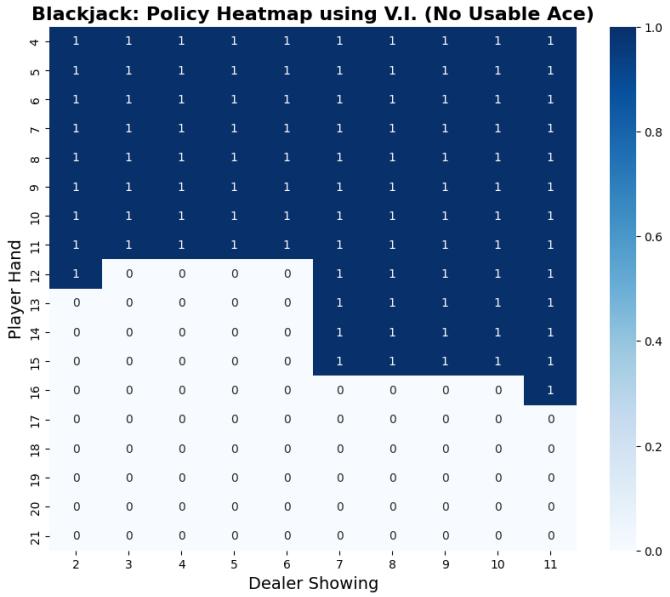

Fig. 3. Blackjack: Policy Heatmap with Hard Hand

Figure 3 and Figure 4 present the policies learned by Value Iteration for Blackjack without and with a usable ace, respectively. These policies represent optimal strategies, with Policy Iteration yielding identical results for $\gamma = 0.6$. The

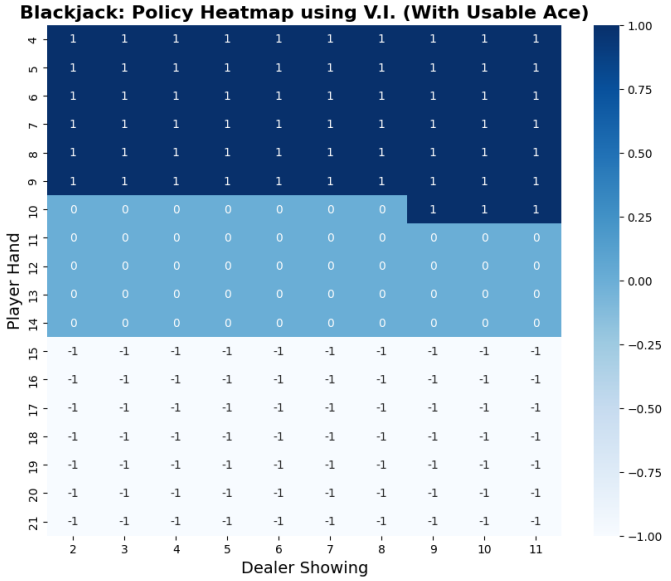

Fig. 4. Blackjack: Policy Heatmap with Soft Hand

player consistently selects "Hit" (1) for hand values between 4 and 16, and transitions to "Stick" (0) as the hand approaches 21. Negative values (-1) indicate invalid situations. In contrast, heatmaps generated by model-free algorithms lack the clear decision boundaries observed here, often displaying random values that evidences suboptimal choices. This discrepancy arises from the trial-and-error nature of model-free methods and the additional time required for them to converge to comparable solutions.

E. Analysis of Model-Free Algorithms

In model-free algorithms, two key parameters significantly influence the learning process: α and ϵ.

Learning Rate (α): This parameter controls the weight given to newly acquired knowledge compared to previous knowledge. A higher α leads to faster learning by prioritizing recent updates, but it may result in instability. Conversely, a lower α slows the learning process, providing greater stability but requiring more time to converge.

Exploration Rate (ϵ): This parameter dictates the balance between exploration (trying new actions to discover their rewards) and exploitation (choosing actions that are known to yield high rewards). A high ϵ encourages exploration, while a low ϵ focuses on exploitation. ϵ often decays over time to ensure sufficient exploration in the early stages, gradually shifting toward exploitation as the policy converges.

Figure 5 visualizes the cumulative policy updates for Q-Learning and SARSA under different exploration strategies. It highlights how exploration parameters affect convergence behavior in model-free algorithms. Limited exploration strategies lead to slower policy updates, as both algorithms prioritize exploitation of known policies. Balanced exploration strikes a middle ground, yielding steady updates with moderate ran-

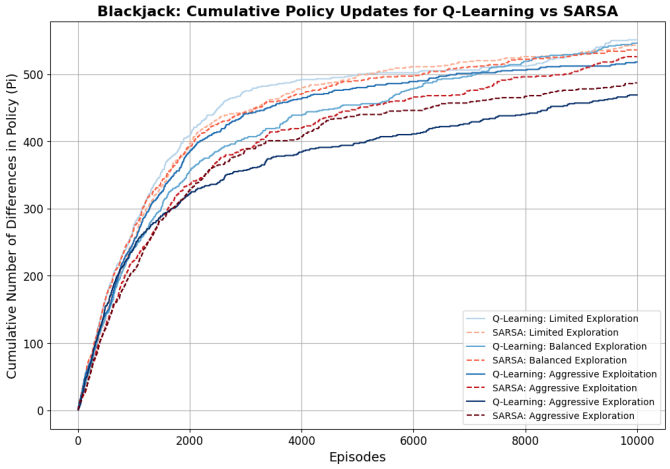

Fig. 5. Blackjack: Exploration-Exploitation in Model-Free Algorithms

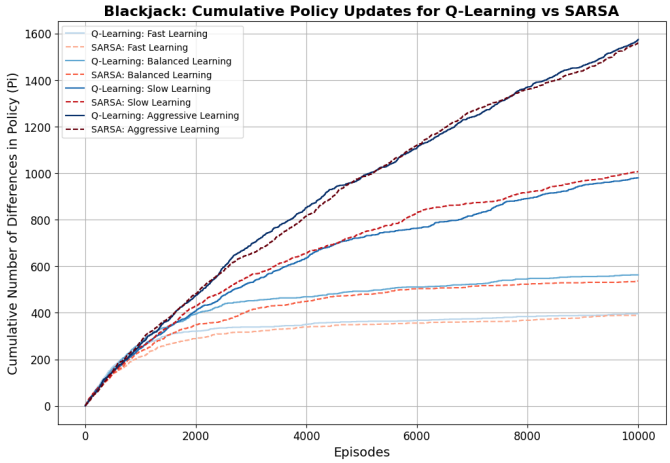

Fig. 6. Blackjack: Learning Strategies in Model-Free Algorithms

domness, while aggressive exploration introduces significant variability in updates due to frequent exploration.

TABLE I
BLACKJACK: RUNTIME FOR EXPLORATION STRATEGIES

Algorithm / Strategy	Limited Exploration	Balanced Exploration	Aggressive Exploitation	Aggressive Exploration
Q-Learning	2.176	2.0659	1.7585	2.0857
SARSA	3.6418	3.0973	2.1479	2.5578

Table I presents the runtime for these strategies. Q-Learning consistently exhibits faster runtimes compared to SARSA across all exploration strategies. Aggressive exploitation has the shortest runtime for both algorithms, as fewer exploratory actions reduce computational overhead.

TABLE II
BLACKJACK: AVERAGE REWARD FOR EXPLORATION STRATEGIES

Algorithm / Strategy	Limited Exploration	Balanced Exploration	Aggressive Exploitation	Aggressive Exploration
Q-Learning	-0.0695	-0.074	-0.0716	-0.0864
SARSA	-0.0626	-0.0723	-0.0621	-0.0781

Table II outlines the average rewards for the same strategies. SARSA slightly outperforms Q-Learning in terms of average reward across most strategies, particularly under limited exploration and aggressive exploitation. However, both algorithms show suboptimal rewards under aggressive exploration, where excessive randomness undermines policy convergence.

These results suggest that balanced exploration provides a suitable trade-off between runtime, reward, and convergence stability, with SARSA showing a slight advantage in reward stability over Q-Learning.

Figure 6, Table III, and Table IV provide insights into the effect of different learning strategies on the performance of model-free algorithms in the Blackjack problem.

Figure 6 compares cumulative policy updates for Q-Learning and SARSA across different learning strategies in the Blackjack environment. Q-Learning consistently shows higher updates due to its off-policy nature, which emphasizes exploration, while SARSA, being on-policy, results in fewer updates due to its conservative approach. ”Slow Learning” leads to the fewest updates, reflecting smoother convergence, whereas ”Fast Learning” and ”Aggressive Learning” exhibit higher updates due to larger initial learning rates or slower decay. Overall, Q-Learning adapts more aggressively, while SARSA offers steadier convergence, highlighting their differing exploratory behaviors.

TABLE III
BLACKJACK: RUNTIME FOR LEARNING STRATEGIES

Algorithm / Strategy	Fast Learning	Balanced Learning	Slow Learning	Aggressive Learning
Q-Learning	2.1004	2.121	2.0736	2.1082
SARSA	2.5001	2.523	2.4608	2.4834

TABLE IV
BLACKJACK: AVERAGE REWARD FOR LEARNING STRATEGIES

Algorithm / Strategy	Fast Learning	Balanced Learning	Slow Learning	Aggressive Learning
Q-Learning	-0.0922	-0.0694	-0.0577	-0.0967
SARSA	-0.0842	-0.0744	-0.0615	-0.0918

Table III shows that runtimes remain consistent across all learning strategies for both algorithms, with Q-Learning being slightly faster than SARSA due to its off-policy nature and independent updates. Interestingly, ”Slow Learning” produces the shortest runtimes for both algorithms, as the slower decay in the learning rate might contribute to more stable updates, reducing the overall computational cost.

Table IV highlights the impact of learning strategies on the average reward. For both algorithms, ”Slow Learning” achieves the best rewards, suggesting that slower decay in the learning rate provides more stable convergence to optimal policies. In contrast, ”Fast Learning” and ”Aggressive Learning” lead to lower rewards, likely due to overcompensation or insufficient exploration caused by aggressive parameter settings. Q-Learning slightly outperforms SARSA in reward outcomes across all strategies, although the differences are marginal for ”Balanced” and ”Slow Learning.”

Overall, the results demonstrate that ”Slow Learning” offers the most effective trade-off between runtime and reward performance, while aggressive strategies may lead to suboptimal

outcomes in Blackjack due to the limited state space and the need for stable convergence.

F. Combined Results

In this subsection, model-based and model-free algorithms are evaluated and compared in terms of effectiveness and runtime to identify the best-performing approach. To ensure a fair comparison, a set of discount factors is applied consistently across all models, and no significant convergence issues were observed during the experiments.

TABLE V
BLACKJACK: RUNTIME FOR DIFFERENT DISCOUNT FACTORS

Algorithm / Gamma	0	0.05	0.2	0.4	0.6	0.8	1
Value Iteration	0.011	0.0221	0.0295	0.0288	0.0291	0.0259	0.0294
Policy Iteration	0.0115	0.0305	0.0329	0.0336	0.0388	0.0408	0.0466
Q-Learning	2.0956	2.0487	2.0717	2.0704	2.0731	2.052	2.0536
SARSA	2.5032	2.4231	2.4554	2.4944	2.4244	2.4094	2.4199

Table V provides a comparison of runtime in the Blackjack problem. Model-based algorithms show significantly shorter runtimes compared to model-free algorithms. The runtime of VI and PI remains consistent across discount factors, reflecting their efficiency in solving this relatively small MDP with 290 states. Conversely, Q-Learning and SARSA exhibit higher runtimes due to their trial-and-error exploration, with SARSA generally taking slightly longer than Q-Learning due to its on-policy nature.

TABLE VI
BLACKJACK: AVERAGE REWARD FOR DIFFERENT DISCOUNT FACTORS

Algorithm / Gamma	0	0.05	0.2	0.4	0.6	0.8	1
Value Iteration	-0.04643	-0.04819	-0.0505	-0.04364	-0.05184	-0.04386	-0.04384
Policy Iteration	-0.04344	-0.0513	-0.04816	-0.04379	-0.04975	-0.04828	-0.04399
Q-Learning	-0.07027	-0.0643	-0.06422	-0.0758	-0.0661	-0.07945	-0.0736
SARSA	-0.0726	-0.0716	-0.0579	-0.081	-0.0804	-0.0726	-0.0877

Table VI highlights the average rewards achieved by the algorithms. Model-based algorithms outperform model-free algorithms, yielding less negative rewards across all values of γ. Among model-based methods, VI and PI perform similarly, with minor variations likely due to their differing computational approaches. Q-Learning and SARSA show relatively worse performance, with SARSA struggling at higher discount factors, likely due to its tendency to follow suboptimal on-policy paths. This indicates that model-based algorithms are better suited for Blackjack, as they converge faster and produce more consistent rewards. V.I. is marginally better than P.I. based on the experiments, however, this algorithm could have been favored by randomness.

III. "LARGE" MDP PROBLEM: CART POLE

A. Problem Description

The Cart Pole problem, as described by Barto, Sutton, and Anderson [7], involves balancing a pole attached to a cart that moves along a frictionless track. The objective is to apply forces to the cart to prevent the pole from falling while ensuring the cart remains within a specified range. This problem is particularly interesting for this report for several reasons: 1) it provides a simple yet impactful real-world scenario that highlights the effectiveness of RL algorithms in solving practical control problems; 2) it features continuous observable states, presenting a unique and challenging domain for RL algorithms traditionally designed for discrete state spaces; 3) its continuous nature requires state discretization through binning, resulting in a "large" number of effective states and offering an opportunity to evaluate algorithm scalability; and 4) it has the potential to uncover convergence challenges, which are critical for analysis and discussion within this report.

State Space: The environment has continuous observable states represented as a vector of four variables:

- Cart position: $x \in (-2.4, 2.4)$
- Cart velocity: $\dot{x} \in (-\infty, \infty)$
- Pole angle: $\theta \in (-12^\circ, 12^\circ)$
- Pole angular velocity: $\dot{\theta} \in (-\infty, \infty)$

Although cart positions and pole angles can extend beyond these ranges, the episode terminates when $x \notin (-2.4, 2.4)$ or $\theta \notin (-12^\circ, 12^\circ)$.

Action Space: The agent selects one of two discrete actions:

- 0: Push the cart to the left.
- 1: Push the cart to the right.

The applied force and its effect on the cart's velocity depend on the current pole angle and its center of gravity.

Starting State: All state variables are initialized to random values in the range $(-0.05, 0.05)$.

Rewards: The environment rewards the agent for balancing the pole. $+1$ is awarded for every time step the pole remains balanced, up to a maximum of 500 steps.

Episode Termination:

- **Pole Angle:** Termination occurs if $|\theta| > 12^\circ$.
- **Cart Position:** Termination occurs if $|x| > 2.4$.
- **Truncation:** Episodes end after 500 steps.

B. Hypothesis

For the cart pole problem, which involves continuous observable states, it is hypothesized that model-free algorithms, such as Q-Learning and SARSA, will outperform model-based algorithms. This is due to the complexity of discretizing the state space for model-based methods, which can lead to suboptimal policies and increased computational demands. Model-free methods, by directly learning from interactions with the environment, are better suited to handle the continuous nature of the problem despite potential convergence issues or extended runtimes.

Between Q-Learning and SARSA, Q-Learning is expected to provide better results due to its off-policy nature, which enables it to explore more aggressively and identify optimal actions even in less-visited states. SARSA, being an on-policy algorithm, is likely to yield more conservative policies and may be less effective at optimizing performance in this challenging environment.

C. Experimental Methodology

The experiments for the Cart Pole problem follow a structure similar to the Blackjack problem but incorporate adjustments to address its continuous observable state space. To

discretize the environment, the Cart Pole wrapper from the `bettermdptools` is used with these parameters: position bins=10, velocity bins=10, angular velocity bins=10, angular center resolution=0.1, angular outer resolution=0.2.

The number of discrete observable states is calculated as: Total States = Position Bins x Velocity Bins x Angular Velocity Bins x Angular Bins.

$$\text{Total States} = 10 \times 10 \times 10 \times 25 = 25,000$$

The discretization introduces a "large" state space, making this problem a unique test case for RL algorithms.

For the model-based algorithms, Value Iteration and Policy Iteration are run across a range of discount factors, $\gamma = [0, 0.05, 0.2, 0.4, 0.6, 0.8, 0.95]$, using a convergence criterion of $\theta = 1 \times 10^{-10}$. Gamma of 1.0 led to convergence/truncation issues and was not included. Metrics collected include convergence plots, training time, and average rewards over 100,000 episodes.

For the model-free algorithms, Q-Learning and SARSA are evaluated under the same range of discount factors ($\gamma = [0, 0.05, 0.2, 0.4, 0.6, 0.8, 0.95]$), with a convergence criterion of $\theta = 1 \times 10^{-10}$. Gamma of 1.0 also presented convergence/truncation issues in these models. Both algorithms are trained for 20,000 episodes to ensure adequate convergence in most cases. The following parameter values are used for learning rate and exploration strategies:
Learning rate: $\alpha_{\text{init}} = 0.5$, $\alpha_{\text{min}} = 0.01$, $\alpha_{\text{decay}} = 0.5$.
Exploration: $\epsilon_{\text{init}} = 1.0$, $\epsilon_{\text{min}} = 0.1$, $\epsilon_{\text{decay}} = 0.9$.
Metrics such as convergence plots, training time, and average rewards after 100,000 iterations are recorded for both algorithms.

Additionally, two supplementary experiments are conducted for the model-free algorithms to analyze parameter sensitivity, keeping the discount factor fixed at $\gamma = 0.6$: 1) exploration-exploitation trade-off with different exploration epsilon parameters, and 2) learning strategies with different learning rate configurations with the alpha parameters. These are the same parameters as were described in the Blackjack Experimental Methodology subsection.

D. Analysis of Model-Based Algorithms

Figure 7 illustrates the convergence of V.I. for the Cart Pole problem under different discount factors. Convergence is measured by the maximum change in state values ($V(s)$) at each iteration. Higher discount factors (γ) such as 0.95 requires 452 iterations to meet the convergence threshold (10^{-10}), reflecting the challenge of propagating long-term rewards. In contrast, smaller values of γ converge much faster as they focus on immediate rewards, requiring fewer updates to stabilize.

Figure 8 depicts the convergence of Policy Iteration P.I. using the number of differences in policy (π) between successive iterations. Regardless of the discount factor, P.I. converges rapidly, with most policies stabilizing within a few iterations, from 2 to 14. The oscillations seen in the early iterations are indicative of the large state space, but the algorithm's ability to quickly stabilize reflects its robustness even for complex environments like the Cart Pole.

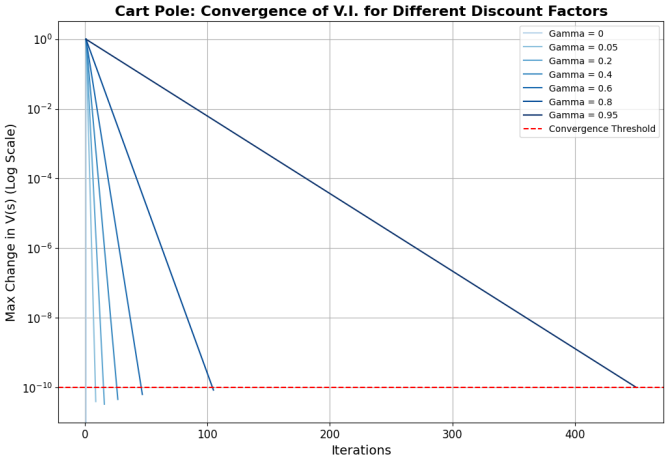

Fig. 7. Cart Pole: Convergence of Value Iteration

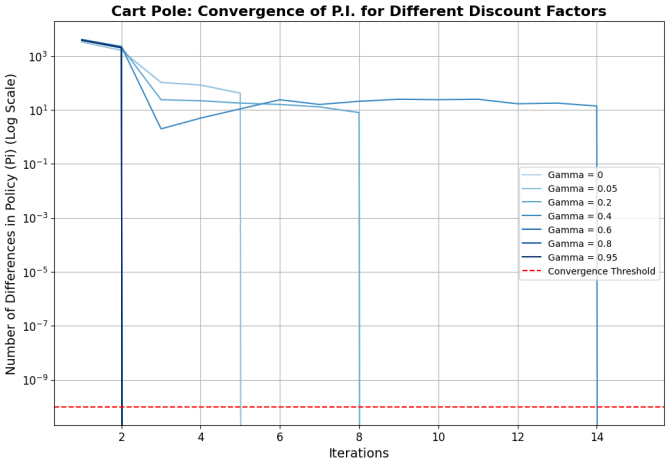

Fig. 8. Cart Pole: Convergence of Policy Iteration

E. Analysis of Model-Free Algorithms

Figure 9 shows the cumulative policy updates for Q-Learning and SARSA under different exploration strategies. Both algorithms demonstrate an increase in policy updates as episodes progress, with SARSA generally experiencing more updates than Q-Learning. Aggressive exploitation strategies converge faster due to reduced exploration, while balanced and limited exploration strategies exhibit slower convergence but lead to more consistent updates.

TABLE VII
CART POLE: RUNTIME FOR EXPLORATION STRATEGIES

Algorithm / Strategy	Limited Exploration	Balanced Exploration	Aggressive Exploitation	Aggressive Exploration
Q-Learning	356.4569	333.2792	171.1998	316.902
SARSA	338.7338	313.9488	165.6561	308.9113

Table VII presents the runtime for different exploration strategies. Aggressive exploitation significantly reduces runtime for both algorithms due to fewer exploratory updates.

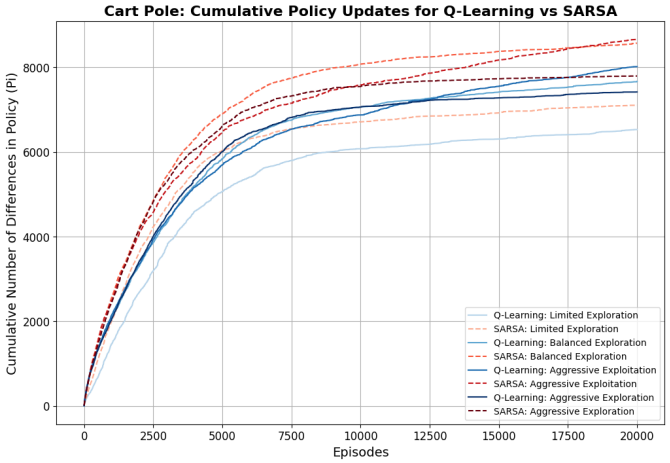

Fig. 9. Cart Pole: Exploration-Exploitation in Model-Free Algorithms

SARSA tends to require slightly more runtime than Q-Learning across all strategies, aligning with its step-by-step policy evaluation nature.

TABLE VIII
CART POLE: AVERAGE REWARD FOR EXPLORATION STRATEGIES

Algorithm / Strategy	Limited Exploration	Balanced Exploration	Aggressive Exploitation	Aggressive Exploration
Q-Learning	103.8695	114.8886	118.8275	98.8163
SARSA	102.6966	104.9405	106.6405	101.2012

Table VIII summarizes the average rewards achieved by each algorithm under different exploration strategies. Aggressive exploitation achieves the highest reward for Q-Learning, but balanced exploration strategies yield competitive rewards with better consistency. SARSA achieves comparable rewards but generally performs slightly lower than Q-Learning, especially under aggressive strategies. In conclusion, Q-Learning demonstrates better performance and reward optimization in the Cart Pole problem, particularly under aggressive and balanced exploration strategies. However, SARSA achieves smoother policy updates with marginally higher runtime costs.

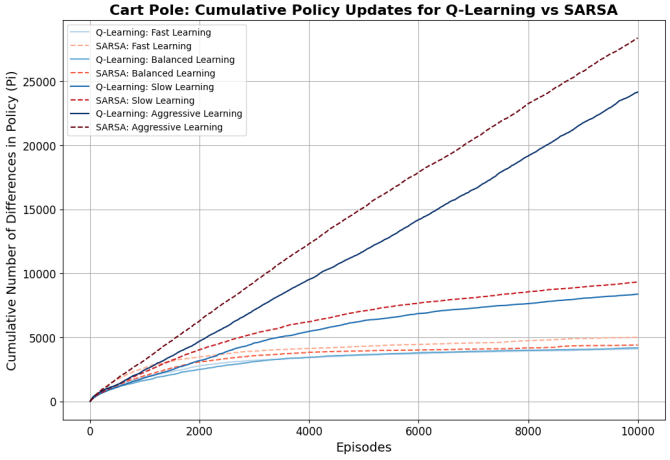

Fig. 10. Cart Pole: Learning Strategies in Model-Free Algorithms

Figure 10 shows the cumulative policy updates for Q-Learning and SARSA under different learning rate strategies. Aggressive learning strategies exhibit the highest number of updates, reflecting rapid changes in policy due to high initial learning rates and slower decay. In contrast, balanced and slow learning strategies lead to more stable policy updates over time, especially in SARSA.

TABLE IX
CART POLE: RUNTIME FOR LEARNING STRATEGIES

Algorithm / Strategy	Fast Learning	Balanced Learning	Slow Learning	Aggressive Learning
Q-Learning	296.9587	303.6578	314.1894	398.9913
SARSA	282.3776	276.0038	320.9455	361.0225

Table IX outlines the runtime of learning strategies. Aggressive learning significantly increases runtime for both Q-Learning and SARSA due to the higher frequency of policy updates and slower convergence. Balanced learning strategies, however, achieve the lowest runtime for both algorithms, demonstrating a good trade-off between stability and performance.

TABLE X
CART POLE: AVERAGE REWARD FOR LEARNING STRATEGIES

Algorithm / Strategy	Fast Learning	Balanced Learning	Slow Learning	Aggressive Learning
Q-Learning	91.5966	100.5694	121.6863	201.6467
SARSA	94.9689	88.5315	101.9854	158.8102

Table X presents the average rewards achieved under different learning strategies. Aggressive learning yields the highest rewards for both Q-Learning and SARSA, reflecting their ability to adapt policies rapidly in the Cart Pole problem. However, Q-Learning consistently outperforms SARSA across all strategies, particularly under aggressive learning, where its cumulative reward is significantly higher. Slow learning strategies achieve stable yet moderate rewards, highlighting their potential in environments with slower dynamics. In conclusion, while aggressive learning maximizes rewards, it comes at the cost of higher runtime. Balanced learning provides a more efficient alternative, especially for SARSA, and demonstrates a good balance between policy stability and runtime efficiency.

F. Combined Results

Table XI and Table XII present the runtime and average rewards achieved by model-based and model-free algorithms for the Cart Pole problem under various discount factors.

TABLE XI
CART POLE: RUNTIME FOR DISCOUNT FACTORS

Algorithm / Gamma	0	0.05	0.2	0.4	0.6	0.8	0.95
Value Iteration	6.4494	7.013	7.101	7.502	8.3904	11.5405	26.2441
Policy Iteration	6.6409	6.7576	7.5653	8.1695	8.74	10.4527	27.6202
Q-Learning	87.4034	161.6077	206.0191	254.3999	293.9535	318.9367	413.4901
SARSA	117.4588	243.7549	164.0285	243.8595	297.3822	305.9773	532.7435

The runtime analysis in Table XI reveals that model-based algorithms (Value Iteration and Policy Iteration) have significantly lower runtimes compared to model-free algorithms (Q-Learning and SARSA). This is expected as model-free

algorithms involve extensive exploration and updates during training. For both Value Iteration and Policy Iteration, runtime increases with higher discount factors, highlighting the increased computational burden of considering long-term rewards. Among the model-free algorithms, Q-Learning consistently outperformed SARSA in runtime efficiency.

TABLE XII
CART POLE: AVERAGE REWARD FOR DISCOUNT FACTORS

Algorithm / Gamma	0	0.05	0.2	0.4	0.6	0.8	0.95
Value Iteration	9.3531	9.3574	9.3551	9.3589	9.3564	9.3535	9.3573
Policy Iteration	9.3537	9.3558	9.3541	9.3519	9.3553	9.3543	9.3572
Q-Learning	11.0687	64.8051	66.9373	78.0355	101.7594	107.3813	109.8427
SARSA	11.4381	20.9092	64.0428	90.8986	103.5075	105.9194	201.7519

Table XII highlights the performance differences. While model-based algorithms achieve consistent rewards across all discount factors, their average rewards remain suboptimal due to their limited capacity to explore and exploit effectively in the continuous state space. Conversely, model-free algorithms demonstrate significant improvements in rewards as γ increases, with Q-Learning achieving the highest reward at $\gamma = 0.95$. SARSA exhibits a similar trend but achieves slightly lower rewards than Q-Learning. This difference stems from Q-Learning's off-policy nature, allowing it to explore more aggressive strategies compared to the on-policy approach of SARSA. Interestingly, the best reward of 201.75 was achieved with SARSA with the highest discount factor.

Given the nature of this problem with a large number of observable states, model-free algorithms like Q-Learning and SARSA excel in maximizing rewards, especially at higher discount factors. The trade-off between runtime and performance must be considered when selecting an algorithm for the Cart Pole problem.

IV. CONCLUSION

This paper explored the performance of model-based and model-free reinforcement learning algorithms on two MDP problems: Blackjack, a small discrete-state MDP, and Cart Pole, a large continuous-state MDP. Our initial hypotheses were that model-based algorithms would outperform model-free methods in the Blackjack problem due to its small, fully observable state space, and that model-free methods, particularly Q-Learning, would excel in the Cart Pole problem due to their adaptability in continuous and larger state spaces.

The results largely validated these hypotheses. For the Blackjack problem, model-based algorithms (V.I. and P.I.) demonstrated faster convergence and lower runtimes compared to model-free methods. Their performance in terms of average rewards was consistent and superior, with V.I. slightly outperforming P.I., likely due to its direct value updates. Model-free methods, while more computationally intensive, struggled with convergence and delivered suboptimal rewards, particularly under aggressive exploration settings. These results confirm the effectiveness of model-based methods in small, well-defined MDPs.

In the Cart Pole problem, model-free algorithms (Q-Learning and SARSA) outperformed model-based approaches in terms of rewards, particularly at higher discount factors. The continuous nature of the state space posed challenges for model-based methods, which required discretization, leading to computational inefficiencies and less effective policies. Q-Learning consistently achieved the highest rewards, leveraging its off-policy nature for more aggressive exploration. SARSA, while slightly less effective, offered smoother convergence and better stability. The trade-off between runtime and performance was evident, with Q-Learning being preferred for environments where reward maximization is prioritized.

Several insights were gained from testing different exploration-exploitation strategies. Balanced exploration (moderate exploration with decaying ϵ) emerged as a strong performer, achieving a good trade-off between runtime and policy quality. Limited exploration led to slower policy updates and lower rewards due to insufficient exploration, while aggressive exploration resulted in erratic policy updates and suboptimal outcomes due to excessive randomness. Similarly, testing different learning rate strategies provided valuable insights. Slow learning rates, with gradual decay, consistently produced stable convergence and higher rewards in both problems, particularly for SARSA. Conversely, aggressive learning rates resulted in faster updates but often led to instability and suboptimal policies. Balanced learning strategies offered the best compromise, delivering competitive rewards while maintaining reasonable runtime efficiency.

Overall, this study highlights the strengths and limitations of RL algorithms based on problem characteristics and underscores the importance of parameter tuning in model-free methods. For small, discrete problems like Blackjack, model-based methods are preferred for their efficiency and reliability. Conversely, in larger, continuous environments like Cart Pole, model-free methods, particularly Q-Learning, demonstrate superior adaptability and reward optimization. These findings reinforce the importance of selecting algorithms and tuning strategies aligned with the complexity and structure of the problem at hand.

REFERENCES

[1] R. Bellman, "Dynamic Programming," *Princeton University Press*, 1957.
[2] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," *MIT Press*, 2018.
[3] M. Puterman, "Markov Decision Processes: Discrete Stochastic Dynamic Programming," *Wiley*, 1994.
[4] C. J. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
[5] G. A. Rummery and M. Niranjan, "On-Line Q-Learning Using Connectionist Systems," *Technical Report*, University of Cambridge, 1994.
[6] "Blackjack Environment Documentation," Gymnasium, available at: https://gymnasium.farama.org/environments/toy$_text/blackjack/$.
[7] Barto, A., Sutton, R., and Anderson, C., "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, 1983.
[8] "Cart Pole Environment Documentation," Gymnasium, available at: https://gymnasium.farama.org/environments/classic$_control/cart_pole/$.