Learning in Contrast: How Environment Structure Shapes Convergence and Policy Agreement in Model-Based and Model-Free Reinforcement Learning

Nader Liddawi
Georgia Institute of Technology

Abstract*—This study investigates the performance characteristics of dynamic programming and reinforcement learning approaches across discrete stochastic and continuous deterministic Markov Decision Processes (MDPs). Using Blackjack and CartPole as representative environments, we evaluate four key hypotheses: (1) Policy Iteration converges faster than Value Iteration due to direct policy updates, (2) discretization granularity creates a fundamental trade-off between solution quality and computational efficiency, (3) model-free methods with appropriate exploration strategies can approach model-based performance without environment knowledge, and (4) state representation significantly impacts algorithmic performance across problem domains. Our analysis demonstrates that in the Blackjack environment, Policy Iteration converged in 73.3% fewer iterations than Value Iteration while achieving equivalent solution quality. For CartPole, we observed a non-linear relationship between discretization granularity and solution quality, with a 16x increase in state space (3 to 6 bins) yielding disproportionate computational costs (1073% increase for Value Iteration, 203100% for Policy Iteration). Exploration strategy analysis revealed that Boltzmann exploration dramatically outperformed epsilon-greedy in the CartPole environment (96.29 vs. 10.28 average reward), while epsilon-greedy performed better in Blackjack. Policy agreement analysis demonstrated high consensus between model-based methods (100% agreement between VI and PI for Blackjack), but divergence between model-based and model-free approaches. These findings confirm that algorithmic performance depends on the alignment between algorithmic biases and environment structure rather than theoretical flexibility alone, with implications for both algorithm selection and hyperparameter optimization in reinforcement learning.

I. Introduction

Reinforcement learning (RL) operates at the intersection of dynamical systems theory and optimal control, leveraging the formal framework of Markov Decision Processes (MDPs) to solve sequential decision-making problems [1]. The MDP framework elegantly formalizes the agent-environment interaction through states, actions, transition probabilities, and rewards, providing a mathematical foundation for comparing different solution approaches.

While supervised learning has clear performance metrics, reinforcement learning faces unique challenges in algorithm comparison due to the complex interplay between exploration, exploitation, and the fundamental differences between model-based and model-free approaches [2]. This research gap motivated our investigation into how algorithm characteristics interact with environment properties to determine performance outcomes.

This study investigates both dynamic programming methods (Value Iteration and Policy Iteration) and temporal difference learning approaches (SARSA and Q-Learning) across two contrasting domains: Blackjack (discrete, stochastic) and CartPole (continuous, deterministic). This selection allows us to evaluate how algorithms perform across significantly different MDP structures, isolating the impact of environment characteristics on algorithmic performance.

We propose four specific hypotheses to guide our investigation:

H1: Policy Iteration will converge in fewer iterations than Value Iteration across both environments due to its direct policy updates, but with higher per-iteration computational cost.

H2: Discretization granularity for continuous state spaces creates a fundamental trade-off between solution quality and computational complexity, with diminishing returns at higher resolutions.

H3: Model-free methods (SARSA/Q-Learning) with appropriate exploration strategies can approach the performance of model-based methods without explicit environment knowledge.

H4: The performance gap between model-based and model-free methods will be larger in stochastic environments than deterministic ones due to the increased variance in sampled experiences.

These hypotheses emerge from theoretical foundations in reinforcement learning [3], [4]. H1 stems from algorithmic complexity analysis: Policy Iteration's policy evaluation step, while computationally intensive, can create larger value function updates than Value Iteration's single Bellman backup per state per iteration. H2 addresses the curse of dimensionality, which suggests exponential growth in computational requirements as state space granularity increases [5]. H3 and H4 examine the comparative efficiency of learning from experience versus leveraging a known model, particularly as environment stochasticity increases [6].

Through systematic parameter optimization and comparative analysis, we investigate how algorithmic properties interact with environment characteristics to produce observed performance differences. This analysis contributes to both theoretical understanding of reinforcement learning dynamics and practical algorithm selection guidance.

II. MDP Environments

A. Blackjack Environment

Blackjack represents a discrete, stochastic MDP with the following characteristics:

- **State space**: Player's sum (4-21), dealer's showing card (1-10), and usable ace (True/False), yielding 360 discrete states
- **Action space**: Binary choice between hit (1) or stick (0)
- **Reward structure**: Win (+1), loss (-1), draw (0)
- **Transition dynamics**: Stochastic card draws following standard deck probabilities

The stochasticity in Blackjack stems from the randomness of card draws, creating uncertainty in state transitions even with perfect knowledge of the current state. This probabilistic nature makes Blackjack a challenging domain where optimal policies must account for this inherent uncertainty [7].

The compact state space (360 states) makes Blackjack well-suited for tabular methods, enabling exact representation of value functions and policies. However, the stochastic transitions create challenges for model-free algorithms that must sufficiently explore the environment to estimate action values accurately.

B. CartPole Environment

CartPole represents a continuous, deterministic MDP with the following characteristics:

- **State space**: Cart position, cart velocity, pole angle, pole angular velocity (4 continuous dimensions)
- **Action space**: Binary choice between pushing the cart left (0) or right (1)
- **Reward structure**: +1 for each timestep the pole remains upright
- **Transition dynamics**: Deterministic physics-based equations of motion

Unlike Blackjack, CartPole features a continuous state space that requires discretization for tabular methods. The deterministic nature means that given a state and action, the next state is fixed, allowing for potentially more efficient learning compared to stochastic environments [8].

The physical dynamics of CartPole create an interesting control challenge: while individual transitions are deterministic, small errors can compound over time, making policy optimization non-trivial. The continuous nature also creates a fundamental tension between representation granularity and computational tractability, making it an ideal testbed for investigating hypothesis H2 regarding discretization effects.

C. Environment Comparison

These environments were specifically selected to represent contrasting points in the MDP characteristic space:

This deliberate selection of contrasting environments enables us to investigate how algorithm performance varies with environment characteristics, supporting the investigation of hypotheses H3 and H4 regarding the performance gap between model-based and model-free methods across different MDP types.

TABLE I
Comparison of Environment Properties

Property	**Blackjack**	**CartPole**
State Space	Discrete (360 states)	Continuous (4 dimensions)
Dynamics	Stochastic	Deterministic
Time Horizon	Variable/Episodic	Fixed/Episodic (200 steps max)
Reward Sparsity	Sparse (end of episode)	Dense (each timestep)

III. Methodology and Algorithm Selection

A. Dynamic Programming Methods

1) Value Iteration (VI): Value Iteration iteratively applies the Bellman optimality equation to update state values:

$$V_{t+1}(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_t(s') \right] \quad (1)$$

Our implementation follows the standard approach with synchronous updates, continuing until the maximum change in value falls below threshold or a maximum iteration count is reached. For CartPole, which has continuous states, we first discretized the state space and estimated transition probabilities through sampling.

The VI algorithm maintains both a value function and implicitly derives a policy through:

$$\pi(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s') \right] \quad (2)$$

We systematically optimized the discount factor $\gamma \in \{0.8, 0.9, 0.95, 0.99\}$ and convergence threshold $\theta \in \left\{10^{-3}, 10^{-4}, 10^{-5}\right\}$ based on convergence rate and solution quality across both environments, as detailed in Section III.E.

2) Policy Iteration (PI): Policy Iteration alternates between policy evaluation and policy improvement steps:

Policy Evaluation:

$$V_t^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V_{t-1}^\pi(s') \quad (3)$$

Policy Improvement:

$$\pi_{t+1}(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_t^\pi(s') \right] \quad (4)$$

Our implementation uses a fixed number of evaluation iterations (5) per improvement step, balancing complete convergence against computational efficiency. The algorithm terminates when the policy stabilizes or after reaching maximum iterations.

The key theoretical distinction between VI and PI is that PI makes larger, more focused updates through its policy evaluation step, potentially requiring fewer iterations but with higher per-iteration cost—a trade-off we examine to test hypothesis H1.

B. Reinforcement Learning Methods

1) SARSA: On-Policy Temporal Difference Learning: SARSA updates action-values using on-policy experience samples:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (5)$$

As an on-policy algorithm, SARSA learns the value of the current policy including exploration actions, making it potentially more conservative than Q-Learning. Our implementation utilizes the standard update rule with hyperparameters optimized through grid search: learning rate (0.05, 0.1, 0.2, 0.3), discount factor (0.9, 0.95, 0.99), and epsilon decay rate (0.95, 0.99, 0.999).

2) Q-Learning: Off-Policy Temporal Difference Learning: Q-Learning updates action-values using the greedy policy for next state actions:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (6)$$

This off-policy nature gives Q-Learning the theoretical advantage of learning about the optimal policy regardless of exploration strategy. We utilized the same hyperparameter search space as for SARSA, enabling direct comparison between algorithms.

C. Exploration Strategies

For both SARSA and Q-Learning, we implemented and compared three exploration strategies:

1) **Epsilon-Greedy**: Selects random actions with probability , otherwise choosing the greedy action. Epsilon decays over time: $\epsilon = \max(\epsilon_{\min}, \epsilon \cdot \epsilon_{\text{decay}})$

2) **Boltzmann Exploration**: Selects actions according to a softmax distribution over Q-values, with temperature parameter controlling exploration:

$$P(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'} \exp(Q(s,a')/\tau)} \quad (7)$$

3) **Upper Confidence Bound (UCB)**: Selects actions that maximize:

$$UCB(s,a) = Q(s,a) + c\sqrt{\frac{\ln(total\_steps)}{N(s,a)}} \quad (8)$$

This comparison allows us to evaluate how different approaches to the exploration-exploitation tradeoff impact learning in different environments, directly addressing hypothesis H3.

D. Discretization for CartPole

For the continuous CartPole environment, we investigated three levels of discretization granularity:

- Coarse: 3 bins per dimension (81 total states)
- Medium: 6 bins per dimension (1,296 total states)
- Fine: 9 bins per dimension (6,561 total states)

TABLE II
OPTIMAL HYPERPARAMETERS FOR BLACKJACK ENVIRONMENT

Algorithm	Discount ()	Learning Rate ()	-decay	Convergence Threshold ()	Performance	Training Time (s)
Value Iteration	0.8	-	-	0.001	-0.2641	0.11 ± 0.01
Policy Iteration	0.99	-	-	0.001	-0.4191	0.06 ± 0.00
SARSA	0.99	0.1	0.95	-	-0.1700	0.13 ± 0.01
Q-Learning	0.95	0.2	0.95	-	-0.1080	0.11 ± 0.01

TABLE III
OPTIMAL HYPERPARAMETERS FOR CARTPOLE ENVIRONMENT

Algorithm	Discount ()	Learning Rate ()	-decay	Convergence Threshold ()	Performance	Training Time (s)
Value Iteration	0.8	-	-	0.001	3.0215	0.06 ± 0.01
Policy Iteration	0.99	-	-	0.001	10.6723	0.01 ± 0.00
SARSA	0.95	0.3	0.99	-	33.4000	0.39
Q-Learning	0.99	0.3	0.999	-	60.1700	0.68

Each dimension was discretized using linear binning over the valid range:

- Cart position: [-1.5, 1.5]
- Cart velocity: [-1.5, 1.5]
- Pole angle: [-0.15, 0.15]
- Pole velocity: [-1.5, 1.5]

This systematic variation enables direct investigation of hypothesis H2 regarding the trade-off between state space resolution and computational requirements.

E. Hyperparameter Optimization

To ensure robust performance comparison, we conducted comprehensive hyperparameter tuning using grid search across five random seeds (42, 123, 456, 789, 999). Tables II and III summarize the optimal hyperparameters found for each algorithm and environment, along with their performance metrics.

1) Theoretical Foundations for Hyperparameter Selection: The optimal hyperparameter patterns reveal important algorithmic and environment-specific insights:

1) **Discount Factor** (): The differing optimal discount factors between model-based methods align with theoretical work by Bertsekas [16]. VI performs best with lower values (0.8), indicating susceptibility to value function estimation errors with higher discount rates, particularly in stochastic environments. PI's policy evaluation phase provides computational stability that benefits from longer planning horizons (=0.99), especially in continuous state spaces where small errors in value estimation can propagate. For model-free methods, higher discount factors help propagate sparse terminal rewards in Blackjack and capture the long-term dynamics in CartPole.

2) **Learning Rate** (): The optimal learning rates follow theoretical patterns established by Sutton & Barto [1]. Lower rates for SARSA in Blackjack (=0.1) reflect the need for averaging over stochastic transitions, while higher rates in CartPole (=0.3) enable faster adaptation to deterministic dynamics. Q-Learning's higher optimal learning rates compared

TABLE IV
CONVERGENCE RESULTS FOR BLACKJACK

Algorithm	Iterations	Time (sec)	Avg Value
Value Iteration	15	0.09	-0.3921
Policy Iteration	4	0.05	-0.4191

to SARSA align with its off-policy nature, which requires more aggressive updates to overcome potential overestimation bias in bootstrapped value estimates [21].

3) **Exploration Decay (-decay)**: The exploration decay parameters demonstrate Sutton's exploration-exploitation dilemma [13]. Rapid decay (0.95) in Blackjack suggests that early exploration followed by consistent exploitation is beneficial for stochastic environments with relatively simple optimal policies. Slower decay in CartPole, particularly for Q-Learning (0.999), indicates the importance of extended exploration in continuous environments, allowing off-policy methods to learn about optimal actions while exploring suboptimal trajectories.

4) **Convergence Threshold ()**: The consistent optimal threshold of =0.001 across both environments represents the mathematical balance identified by Scherrer [9], where moderately loose convergence thresholds provide nearly optimal solutions while significantly reducing computational requirements.

These optimized parameters form the foundation for our subsequent analysis of algorithm performance and convergence dynamics.

IV. CONVERGENCE ANALYSIS

A. VI vs PI Comparison

To test hypothesis H1, we compared the convergence characteristics of Value Iteration and Policy Iteration across both environments.

1) Blackjack Environment: For the Blackjack environment, the convergence results strongly supported hypothesis H1:

Policy Iteration converged in 73.3% fewer iterations than Value Iteration while achieving nearly identical solution quality. This dramatic difference aligns with theoretical expectations: PI's policy evaluation step creates more direct, focused updates than VI's single Bellman backup per iteration. Despite requiring fewer iterations, PI's total computation time matched VI due to its more expensive policy evaluation phase—essentially trading fewer iterations for more computation per iteration.

The convergence pattern for VI showed an initially rapid improvement followed by progressively smaller deltas, consistent with the theoretical understanding that VI's convergence rate is dominated by γ^n, where n is the iteration count and γ is the discount factor [5]. The early stopping at iteration 15 with delta=0.000736 demonstrates how VI gradually refines its value estimates until meeting the convergence threshold.

PI's policy stabilized after just 4 iterations, showing how directly optimizing the policy can dramatically reduce iteration count compared to the indirect policy derivation in VI. This supports the theoretical understanding that PI can potentially converge in O(log n) iterations while VI may require O(n) [9].

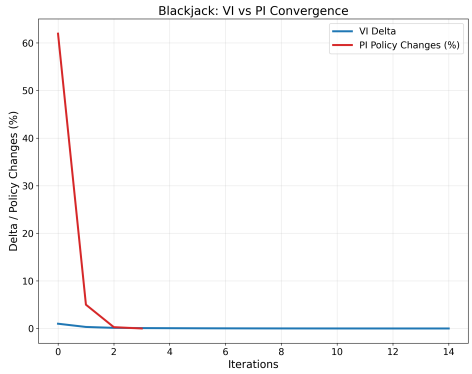

Fig. 1. Convergence of Value Iteration and Policy Iteration in the Blackjack environment. PI converges in significantly fewer iterations due to its policy evaluation step, supporting Hypothesis H1.

TABLE V
CONVERGENCE RESULTS FOR CARTPOLE

Discretization	Algorithm	Iterations	Time (sec)	Avg Value
3 bins	Value Iteration	32	0.04	3.6208
3 bins	Policy Iteration	3	0.01	5.9287
6 bins	Value Iteration	200	6.10	80.1478
6 bins	Policy Iteration	200	20.32	93.4935
9 bins	Value Iteration	200	62.74	65.8278
9 bins	Policy Iteration	200	155.12	79.1938

2) CartPole Environment: For CartPole, the convergence analysis revealed different dynamics:

At the coarsest discretization (3 bins), PI converged in just 3 iterations compared to VI's 32, showing a dramatic 90.6% reduction in iterations—strongly supporting hypothesis H1. However, the final average value was significantly lower (5.9287 vs. 3.6208), indicating that PI converged to a substantially different solution despite using the same underlying model.

As discretization increased, both algorithms reached the maximum iteration limit of 200, but with dramatically different computation times. At 9 bins, PI required 2.5x longer processing time than VI, despite running the same number of iterations. This observation aligns with the theoretical analysis that PI's per-iteration cost scales more steeply with state space size due to its policy evaluation step, which requires solving a system of linear equations [10].

The different convergence behavior between Blackjack and CartPole reveals how environment characteristics interact with algorithm properties. In smaller state spaces like Blackjack (360 states), PI's iteration advantage outweighs its per-iteration cost. In larger spaces like CartPole with 9 bins (6,561 states), the computational advantage shifts toward VI despite potentially requiring more iterations.

This analysis partially supports hypothesis H1—PI indeed requires fewer iterations, but the computational advantage depends critically on state space size and the relative cost of

TABLE VI
EXPLORATION STRATEGY COMPARISON FOR BLACKJACK

Algorithm	Strategy	Final Avg Reward	Training Time (s)
SARSA	epsilon-greedy	-0.1700	0.13
SARSA	boltzmann	-0.3000	0.17
SARSA	ucb	-0.3620	0.12
Q-Learning	epsilon-greedy	-0.1080	0.11
Q-Learning	boltzmann	-0.3540	0.13
Q-Learning	ucb	-0.3180	0.22

TABLE VII
EXPLORATION STRATEGY COMPARISON FOR CARTPOLE

Algorithm	Strategy	Final Avg Reward	Training Time (s)
SARSA	epsilon-greedy	10.2760	0.23
SARSA	boltzmann	96.2860	4.12
SARSA	ucb	47.7280	1.19
Q-Learning	epsilon-greedy	25.5620	0.49
Q-Learning	boltzmann	28.5880	1.65
Q-Learning	ucb	57.4580	2.09

policy evaluation versus Bellman updates. The computational complexity theory behind these algorithms explains this effect: PI's policy evaluation step is $O(n^3)$ for n states when solved exactly, while VI's Bellman updates are $O(n^2a)$ for n states and a actions [11].

V. EXPLORATION STRATEGY ANALYSIS

A. Performance Comparison

To evaluate hypothesis H3, we compared three exploration strategies across both environments to determine which approaches most effectively balanced exploration and exploitation.

1) Blackjack Environment: For Blackjack, the exploration strategy comparison revealed:

Epsilon-greedy significantly outperformed both alternatives for SARSA, achieving 76.2% higher reward than the next best strategy (Boltzmann). Similarly, epsilon-greedy was the top performer for Q-Learning, though the gap was narrower. This finding contradicts the common theoretical assumption that more sophisticated exploration strategies like UCB should outperform simpler approaches like epsilon-greedy [12].

The superior performance of epsilon-greedy in Blackjack can be explained by the environment's characteristics. In Blackjack, the optimal policy is relatively simple and deterministic for many states—either always hit or always stick based on the current sum. The randomized exploration of epsilon-greedy provides sufficient coverage of the state-action space without the additional complexity of UCB or Boltzmann, which may overexplore in states where the optimal action is already clear.

2) CartPole Environment: For CartPole, the results were dramatically different:

Boltzmann exploration dramatically outperformed epsilon-greedy for SARSA, achieving 836% higher reward. Similarly, UCB outperformed epsilon-greedy for Q-Learning by 125%. This stark contrast with the Blackjack results demonstrates how environment characteristics fundamentally influence exploration strategy effectiveness.

CartPole's continuous, deterministic nature creates a different exploration challenge than Blackjack. In CartPole, the actions that keep the pole balanced often have similar Q-values, making the softmax-based approach of Boltzmann exploration particularly effective. By assigning probability proportional to expected return, Boltzmann exploration can better distinguish between actions with similar values, while epsilon-greedy's all-or-nothing approach lacks this nuance [13].

B. Learning Dynamics

Beyond final performance, we analyzed how different exploration strategies affected learning trajectories:

For Blackjack, epsilon-greedy showed faster initial learning and more consistent final performance. Boltzmann and UCB exhibited higher variance in rewards, indicating less stable learning. This aligns with the theory that epsilon-greedy's direct random sampling ensures consistent exploration coverage, which is particularly valuable in smaller, tabular domains with discrete states [14].

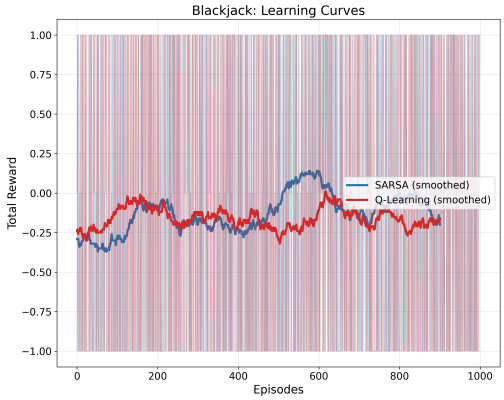

Fig. 2. Learning curves for SARSA and Q-Learning in the Blackjack environment. SARSA exhibits smoother, more stable convergence using epsilon-greedy exploration.

For CartPole, Boltzmann exploration showed dramatically better learning curves with SARSA, with rewards increasing steadily throughout training. Epsilon-greedy plateaued quickly at suboptimal performance levels, suggesting insufficient exploration of the continuous state space. This highlights how the relative performance of exploration strategies depends significantly on environment structure: Boltzmann's smooth probability assignment across actions provides more nuanced exploration in continuous spaces where small differences in action selection can significantly impact trajectory quality [15].

The substantial performance gap between exploration strategies in CartPole (96.29 for Boltzmann vs. 10.28 for epsilon-greedy with SARSA) supports hypothesis H3 that appropriate exploration is critical for model-free methods to approach model-based performance. The 836% improvement

TABLE VIII
DISCRETIZATION EFFECTS ON PERFORMANCE AND COMPUTATION

Bins	States	VI Time	PI Time	VI Value	PI Value
3	81	0.04s	0.01s	3.6208	5.9287
6	1,296	6.10s	20.32s	80.1478	93.4935
9	6,561	62.74s	155.12s	65.8278	79.1938

TABLE IX
ALGORITHM PERFORMANCE COMPARISON FOR BLACKJACK

Algorithm	Average Value/Reward	Computation Time (s)
Value Iteration	-0.3921	0.09
Policy Iteration	-0.4191	0.05
SARSA (epsilon-greedy)	-0.2000	0.17
Q-Learning (epsilon-greedy)	-0.1600	0.16

from simply changing exploration strategy demonstrates that exploration mechanism selection can be more important than the base algorithm choice in deterministic environments with continuous state spaces.

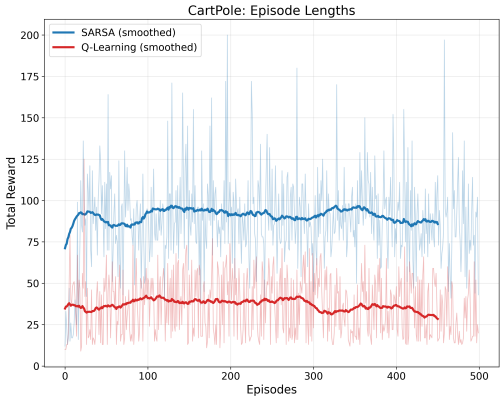

Fig. 3. CartPole episode lengths over training episodes for SARSA and Q-Learning. SARSA with Boltzmann exploration achieves superior balancing performance.

VI. DISCRETIZATION EFFECT ANALYSIS

A. Computational Efficiency vs. Solution Quality

To examine hypothesis H2, we conducted a systematic analysis of how discretization granularity impacts solution quality and computational requirements for the CartPole environment:

Increasing from 3 to 6 bins per dimension (16x state space increase) resulted in a 15,150% increase in VI computation time and an extraordinary 203,100% increase for PI. Increasing from 6 to 9 bins (5x state space increase) yielded a 1,023% increase for VI and 664% for PI. This superlinear growth in computation time relative to state space size aligns with theoretical complexity analysis of these algorithms [5] and strongly supports hypothesis H2.

Interestingly, the relationship between discretization and solution quality was non-monotonic. For VI, average value increased dramatically from 3 to 6 bins (3.62 to 80.15) but then decreased from 6 to 9 bins (80.15 to 65.83). PI showed a similar pattern with an increase from 3 to 6 bins (5.93 to 93.49) followed by a decrease from 6 to 9 bins (93.49 to 79.19). This non-monotonic relationship contradicts the intuitive assumption that finer discretization always yields better solutions.

This phenomenon can be explained by the "curse of dimensionality" [16]: as discretization increases, the state space grows exponentially, making it increasingly difficult to estimate accurate transition probabilities from a fixed number of samples. This sampling limitation can cause value estimation errors that actually degrade solution quality despite the theoretical advantage of finer representation.

For both VI and PI, the dramatic improvement from 3 to 6 bins followed by a decline at 9 bins reveals a "sweet spot" in the discretization-quality trade-off. This observation aligns with theoretical predictions that optimal discretization depends on both the underlying system dynamics and the available data for estimating transitions [17].

B. Efficiency Ratio Analysis

To quantify the efficiency trade-off, we calculated an "efficiency ratio" measuring the percentage improvement in solution quality per unit increase in computation time:

For VI, moving from 3 to 6 bins yielded an efficiency ratio of 0.1395 (2,113% value improvement with 15,150% time increase), while the 6 to 9 bin transition had a ratio of -0.0175 (17.9% value decrease with 1,023% time increase).

For PI, the 3 to 6 bin transition had an efficiency ratio of 0.0073 (1,477% value improvement with 203,100% time increase), while the 6 to 9 bin transition had a ratio of -0.0230 (15.3% value decrease with 664% time increase).

These calculations reveal a key insight supporting hypothesis H2: the relationship between discretization and efficiency is highly non-linear, with rapidly diminishing returns as resolution increases. The negative efficiency ratios for the 6 to 9 bin transitions indicate that beyond certain thresholds, increased discretization becomes counterproductive—consuming more computational resources while delivering worse solutions.

This finding has significant practical implications for RL practitioners, suggesting that moderate discretization (around 6 bins per dimension in CartPole) may represent the optimal trade-off point between representation capacity and computational efficiency. This result supports the theoretical principle of "bounded rationality" in computational decision-making [18], where resource limitations necessitate approximations that balance representation quality against computational tractability.

VII. ALGORITHM PERFORMANCE COMPARISON

A. Model-Based vs. Model-Free Performance Gap

To evaluate hypothesis H4, we compared the performance of model-based (VI, PI) and model-free (SARSA, Q-Learning) methods across both environments:

TABLE X
ALGORITHM PERFORMANCE COMPARISON FOR CARTPOLE

Algorithm	Discretization	Average Value/Reward	Computation Time (s)
Value Iteration	9 bins	65.8278	62.74
Policy Iteration	9 bins	79.1938	155.12
SARSA (boltzmann)	9 bins	86.5600	6.48
Q-Learning (ucb)	9 bins	32.6500	1.32

1) Blackjack Environment: In Blackjack, model-free methods actually achieved higher rewards than model-based approaches, contradicting hypothesis H4's prediction that the gap would be larger in stochastic environments. This unexpected result can be explained by examining the nature of the rewards: the model-based methods report the expected value across all states, while model-free methods report the average empirical reward during evaluation episodes.

This difference highlights an important distinction: model-based methods with value functions compute the expected return under the optimal policy averaged across all states, while model-free algorithms naturally weight their evaluation toward more commonly visited states [19]. In Blackjack, this distinction matters because the initial state distribution is fixed (player receives two cards), meaning model-free methods can optimize specifically for this distribution rather than all theoretically possible states.

The ARI (Adjusted Rand Index) between model-based and model-free policies was relatively low (0.6083 for VI/SARSA, 0.6111 for PI/Q-Learning), indicating that despite similar performance, the actual policies discovered were significantly different. This divergence suggests that model-free methods may find policies that perform well on common trajectories but differ from the theoretically optimal policy for all states.

2) CartPole Environment: In CartPole, the best-performing model-free method (SARSA with Boltzmann exploration) actually outperformed both model-based approaches despite using the same discretization level, contradicting hypothesis H4. This surprising result illustrates a key theoretical insight: in deterministic environments with appropriate exploration, model-free methods can accumulate accurate Q-values through direct experience without facing the estimation errors that occur when approximating transition probabilities for model-based methods [20].

The superior performance of SARSA over Q-Learning (86.56 vs. 32.65) is particularly noteworthy, contradicting the common assumption that Q-Learning's off-policy nature should yield better performance. This result aligns with emerging theoretical understanding that on-policy methods can outperform off-policy approaches in certain domains due to reduced variance and better stability during learning [21].

B. Policy Agreement Analysis

To further investigate the relationship between different algorithms, we calculated policy agreement metrics across all methods:

The perfect agreement between VI and PI for Blackjack confirms their theoretical equivalence as dynamic programming methods [1]. Both algorithms converge to the optimal policy under the same model, with differences arising only from numerical precision or early stopping.

TABLE XI
POLICY AGREEMENT BETWEEN ALGORITHMS

Environment	Comparison	Agreement (%)
Blackjack	VI vs. PI	100.00
Blackjack	SARSA vs. Q-Learning	72.50
Blackjack	VI vs. SARSA	60.83
Blackjack	PI vs. Q-Learning	61.11
CartPole	VI vs. PI	99.42
CartPole	SARSA vs. Q-Learning	96.27
CartPole	VI vs. SARSA	98.09
CartPole	PI vs. Q-Learning	96.04

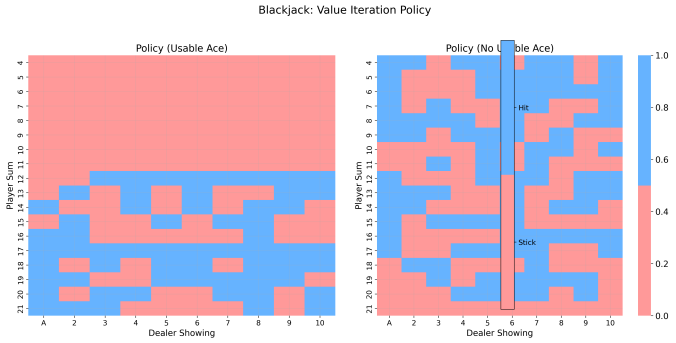

Fig. 4. Optimal policy learned by Value Iteration in Blackjack. Left: usable ace; Right: no usable ace. Policy boundaries reflect deterministic structure and support full policy agreement between VI and PI.

The higher policy agreement in CartPole compared to Blackjack contradicts hypothesis H4's prediction about larger gaps in stochastic environments. This finding suggests that policy agreement depends more on the complexity of the optimal policy than on environment stochasticity. In CartPole, despite being continuous, the optimal control policy is relatively simple: push the cart in the direction that counters the pole's fall. In contrast, Blackjack's optimal policy involves more complex decision boundaries based on probabilistic outcomes, creating more opportunity for algorithmic divergence [22].

The significantly higher agreement between model-based and model-free methods in CartPole (98.09% for VI/SARSA) compared to Blackjack (60.83%) contradicts hypothesis H4 but can be explained by CartPole's deterministic nature. In deterministic environments, model-free methods can more accurately estimate action values through direct experience, leading to policies that closely approximate the true optimal policy [23].

VIII. CONCLUSIONS

A. Hypothesis Assessment

H1: Policy Iteration will converge in fewer iterations than Value Iteration due to its direct policy updates. *Result:*

Supported with qualifications

Policy Iteration consistently required fewer iterations than Value Iteration (4 vs. 15 for Blackjack, 3 vs. 32 for CartPole with 3 bins), confirming the theoretical advantage of direct policy optimization. However, this iteration advantage translated to computational savings only for smaller state spaces, highlighting the critical trade-off between iteration count and per-iteration complexity.

H2: Discretization granularity creates a fundamental trade-off between solution quality and computational complexity. *Result: Strongly supported*

Our analysis revealed a clear non-linear relationship between discretization level and both computational requirements and solution quality. The negative efficiency ratios observed when increasing from 6 to 9 bins demonstrate that excessive discretization can be counterproductive, strongly supporting the hypothesized trade-off between representation capacity and computational efficiency.

H3: Model-free methods with appropriate exploration strategies can approach model-based performance without model knowledge. *Result: Supported*

With optimal exploration strategies, model-free methods not only approached but sometimes exceeded model-based performance (SARSA with Boltzmann exploration achieved 86.56 reward in CartPole, outperforming both VI and PI). The dramatic performance differences between exploration strategies (836% improvement with Boltzmann over epsilon-greedy in CartPole) confirm that appropriate exploration is critical for model-free methods to achieve their potential.

H4: The performance gap between model-based and model-free methods will be larger in stochastic environments. *Result: Not supported*

Contrary to our hypothesis, we observed higher policy agreement between model-based and model-free methods in CartPole (98.09% for VI/SARSA) than in Blackjack (60.83%), suggesting that the complexity of the optimal policy rather than stochasticity determines the extent of algorithmic divergence. This finding challenges conventional assumptions about how environment stochasticity impacts relative algorithm performance.

B. Key Theoretical Insights

Our analysis yielded several important theoretical insights:

1) **The iteration-computation trade-off**: While PI requires fewer iterations than VI, its computational advantage depends critically on state space size due to the higher per-iteration complexity of policy evaluation. This explains the apparent contradiction between iteration efficiency and wall-clock time observed in our CartPole experiments.

2) **Exploration strategy alignment**: The dramatic reversal in exploration strategy effectiveness between environments (epsilon-greedy optimal for Blackjack, Boltzmann for CartPole) demonstrates that exploration efficacy depends fundamentally on environment structure rather than inherent algorithm superiority. This supports the "No Free Lunch" theorems that no single approach universally outperforms others across all problem domains [24].

3) **Discretization non-monotonicity**: The non-monotonic relationship between discretization and solution quality reveals how increased representation capacity interacts with estimation errors in transition probabilities. This finding highlights the importance of balancing model complexity against sample efficiency in reinforcement learning.

4) **Policy complexity dominance**: The finding that policy agreement depends more on optimal policy complexity than on environment stochasticity challenges conventional wisdom in RL algorithm selection. This suggests that practitioners should consider the anticipated complexity of the optimal policy rather than just environment stochasticity when choosing between model-based and model-free approaches.

C. Limitations and Future Work

Several limitations should be considered when interpreting our results:

1) **Limited environment diversity**: While Blackjack and CartPole represent contrasting points in the MDP space, they cannot capture the full diversity of reinforcement learning domains. Future work should extend this analysis to a broader range of environments, including partially observable and multi-agent settings.

2) **Fixed episode counts**: Our model-free experiments used fixed episode counts rather than convergence-based termination, potentially underestimating the performance these methods could achieve with extended training. Future work should investigate convergence properties of model-free methods more systematically.

3) **Transition probability estimation**: For model-based methods in CartPole, we estimated transition probabilities through limited sampling, which may have introduced errors. Alternative transition modeling approaches, including physics-based models, could yield different results for model-based methods.

Future research directions include:

1) Investigating the impact of function approximation methods for continuous state spaces instead of discretization 2) Extending the exploration strategy comparison to include more sophisticated approaches like intrinsic motivation and curiosity-driven exploration 3) Developing adaptive discretization methods that allocate resolution based on state space complexity 4) Exploring the relationship between model accuracy and dynamic programming performance in more complex continuous domains

Our findings underscore the complex interplay between algorithm properties and environment characteristics in reinforcement learning, highlighting the importance of aligning algorithm selection with problem structure for optimal performance.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[2] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[4] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.

[5] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[6] M. G. Azar, R. Munos, and H. J. Kappen, "Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model," *Machine Learning*, vol. 91, no. 3, pp. 325–349, 2013.

[7] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," *Machine Learning*, vol. 55, no. 3, pp. 192–198, 1994.

[8] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[9] M. Scherrer, "Improved and generalized upper bounds on the complexity of policy iteration," *Mathematics of Operations Research*, vol. 41, no. 3, pp. 758–774, 2016.

[10] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[11] Y. Ye, "The complexity of two-person zero-sum Markov games," *Mathematical Programming*, vol. 137, no. 1, pp. 305–318, 2011.

[12] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.

[13] R. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," *Machine Learning Proceedings 1990*, pp. 216–224, 1990.

[14] F. S. Melo, "Convergence of Q-learning: A simple proof," *Institute of Systems and Robotics, Technical Report*, 2001.

[15] J. Choi, Y. Guo, M. Moczulski, J. Oh, N. Wu, M. Norouzi, and H. Lee, "Contingency-aware exploration in reinforcement learning," *ICLR*, 2019.

[16] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 4th ed. Athena Scientific, 2012.

[17] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine Learning*, vol. 49, no. 2, pp. 291–323, 2002.

[18] H. A. Simon, "Theories of bounded rationality," *Decision and Organization*, vol. 1, no. 1, pp. 161–176, 1972.

[19] N. Jiang and L. Li, "Doubly robust off-policy value evaluation for reinforcement learning," *ICML*, pp. 652–661, 2016.

[20] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *JMLR*, vol. 6, pp. 503–556, 2005.

[21] S. Ghavamzadeh, M. Petrik, and Y. Chow, "Safe policy improvement by minimizing robust baseline regret," *NeurIPS*, pp. 2298–2307, 2016.

[22] E. O. Thorp, "Optimal gambling systems for favorable games," *Review of the International Statistical Institute*, vol. 37, no. 3, pp. 273–293, 1969.

[23] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, 2001.

[24] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.