A Comparative Analysis of Expected SARSA and Monte Carlo Methods in Blackjack Strategy Learning

Vidya Padmanabha
Master Artificial Intelligence, Faculty of Computer Science
Technical University of Applied Sciences, Würzburg-Schweinfurt
vidya.padmanabha@study.thws.de

***Abstract*—You walk into a casino and decide to try blackjack, hoping to make some money out of the ordinary. But which strategy helps you beat the dealer? This paper evaluates reinforcement learning methods to determine which approaches help players win more consistently. The paper here helps compare ExpectedSARSA and Monte Carlo algorithms across a handful of scenarios in a realistic blackjack environment featuring multi-deck play, card counting with true count-based betting, and various rule variations whose favor is in the player's hand. All these incorporate adaptive learning rates, epsilon decay, and optimistic initialization, with training from 200,000 to 1,000,000 episodes.**

Results show enhanced algorithms consistently outperform standard implementations. Enhanced ExpectedSARSA achieves superior convergence, while card counting integration demonstrates measurable advantages over basic strategy. Performance is measured through win rates, ROI, and average profit per game.

This research bridges theoretical reinforcement learning with practical gambling applications, providing evidence-based insights into which methods deliver real advantages at the blackjack table. The framework establishes a foundation for evaluating RL algorithms in gaming scenarios, where strategy optimization directly impacts monetary outcomes.

***Index Terms*—Reinforcement Learning, Expected SARSA, Monte Carlo Methods, Blackjack Strategy, Card Counting**

I. Introduction

Blackjack has attracted the attention of players and researchers over the decades, not only because it is popular in the casinos all over the world, but also since it is an extraordinary game between skills and chances. Unlike games in which one can win simply through luck, blackjack cannot be mastered without critical thinking and mathematical accuracy, hence The most suitable experimental environment for intelligence networks. The uncertainty of the result, the opacity of the information, and dynamic choices that are the components of this model create the very type of environment through which the machine learning algorithms can learn through experience to discover optimal decision strategies.

This research aims at exploring, applying, and comparing various reinforcement learning Algorithms to help understand the best move to play in the game of blackjack, i.e, Expected SARSA algorithm and Monte Carlo methods.

The goal of the paper is to test how the outcomes of performance and learning by an agent change under the influence of various strategic levels (whether the basic play, card counting, and certain variations of the rules of the game). This paper shall, in a more systematic way than has yet been done in various situations, ask the question: When, why, and how, does reinforcement learning actually give any advantage to the blackjack player?

A. The Art of Blackjack

The art of any blackjack strategy is simply to know precisely when to hit, stand, split, or whether you are going to bust. Each card also has a numerical value, with numbered cards valued 2 to 10 taken to be equal to their face value, the face cards (Jacks, Queens, and Kings) valued as 10, and Aces valued either as 1 or 11. This potential play of an Ace to a two-card 21 is treated to add an additional wrap of strategy, the chance of having what is termed a soft hand, a hand with an Ace that counts as 11 (and not one or 11).

There are two popular choices available to the players in our application, which are: the decision to hit (to get an extra card) or to stand (to keep the hand as is). There are additional plays even in the real blackjack, e.g., there is a possibility of doubling or splitting, but we tweak our hands to play only on the basis of these basic decisions first, so comparison of the algorithm is much cleaner, as this could still have the strategic part of the game and later introduce two new variations that will be favourable for the player. Further to it, there is also an option of Insurance where the dealer has an Ace as an up card; the player can bet that the unseen card of the dealer is a 10 card and in the case of a concealed 10, then the player gets two times the money which he bets [1]. The dealer is guided by a fixed set of rules and continues to take cards until he gets at least 17, but in this implementation, the usual variation of dealer hits on a soft 17 is taken into consideration.

Blackjack is particularly hard both for people and for algorithms due to a partial information issue. The players must make decisions based only on their hands and the visible card on the dealer (the up-card), but the dealer has a card that is face down (the hole card) until the hand is over. The inability to know the next card to be drawn, coupled with the probabilistic nature of drawing cards when restricted to a fixed deck of cards, combines to form a rich decision scenario that is rewarding in both a statistical and dynamically adjusting nature.

B. Strategic Foundations in Blackjack: Basic Play vs. Card Counting

The traditional blackjack strategy can be classified into two broad areas:

1) Basic Play: The "Zero-memory" or the so-called "Basic" strategies have been deduced as well [2]. These strategies are far less complicated. They suppose that there is one single player who competes against the dealer, and that three cards only are known (the two cards of the player and the up card of the dealer) there is no record maintained on plays that have already happened; thus the name zero memory [1]. It is a set of rules that reduce the house edge when used minimally under regular circumstances; however, it does not adjust to the alteration in deck composition due to cards being dealt.

2) Complete Point Count (CPC): There is then a Complete Point Count (CPC) system, where cards are valued when they appear: low cards (2-6) are +1, high cards (10, J, Q, K, A) are -1, and mid cards (7-9) are neutral. The more it counts, the more the chances of high-value cards being turned over next, which may favour the player. The true count goes one step ahead as it accounts for the decks in play. The system splits this true count into bins (-5, 0, 2, 5+), which it uses to produce discrete strategic states of the learning algorithm. The situation when we have the favourable count (count is equal or greater than 2) is when the agents have an opportunity to raise their betting, and with the escalating bets from 1 unit at low counts and 4 units at counts of 4 or more, such an integration of counting and betting strategy is one of the most important innovations of winning strategy.

C. Reinforcement Learning in Blackjack: Bridging Theory to Application

Reinforcement learning presents an entirely different strategy for developing strategies. Rather than memorizing a set of pre-coded rules, RL agents explore their experiences through trial-and-error and refine their actions accordingly with respect to observed results. Such a paradigm of learning is especially suitable for blackjack, whose optimal decisions rely on the interaction of many factors.

The paper concentrates on two kinds of RL methods. Expected SARSA (State-Action-Reward-State-Action), which is an on-policy temporal-difference algorithm that learns while trailed by its actual policy, similar to Q-learning, but in place of it we have it use the expected value on the maximum of pairs that are in the current policy [1]. On the other hand, Monte Carlo methods wait until episodes are done before updating value estimates based on actual returns, getting an unbiased value, but need the full episode end.

The paper describes the system, which refers to game states using a few important elements: the number of cards in the hand of the player, the presence of a usable Ace, the visible up-card of the dealer and in the case of card counting a current bin of the correct count. Depending on the final hand, the only allowed action is either hit (1) or stand (0), and the reward is given in relation to the eventual hand status; i.e., wins lead to positive rewards that are scaled to the bet size at hand, and losses and ties lead to negative rewards, and zero rewards, respectively.

This exploration-exploitation dilemma is obvious in the implementation where there has been an epsilon-greedy action selection. Agents must compromise between attempting the best they think and periodically attempting other options (exploration) to make sure they do not overlook better strategies. The improved agents are sophisticated parameter scheduling, where epsilon and the learning rates diminish with time to move into the exploitation phase as learning proceeds.

The research questions guiding this study will be based on the following: What are the comparisons of Expected SARSA and Monte Carlo algorithms in learning blackjack strategies? What performance advantages do the provisions of a comprehensive point count system deliver? What are the impacts on learning and end-strategy quality of specific rule variations (e.g., resplitting aces, visible dealer hole cards) on their effects? Does it mean that adaptive parameter versions of algorithms can outclass their regular versions?

II. Methodology

A. Theoretical Basis

The reinforcement learning strategy learning approach to Blackjack strategy learning gives it the form of a Markov Decision Process (MDP) under which an agent learns good policies by interacting with the environment. The tuple (S, A, P, R, gamma) is as follows: the states S contains hand sets of players and deck structure; the set of actions A = 0, 1 denotes stand and hit actions; the probability of transition P is based on deck structure and game settings, the benefit of the outcome R is monetary, and discount factor weights future rewards. Its basic goal is to acquire the best action value function Q * (s, a) that will yield the maximum expected cumulative reward. This is an optimal Bellman equation.

$$Q(s,a) = \mathbb{E}\left[R(s,a) + \gamma \max_{a'} Q(s',a')\right] \tag{1}$$

[3] Two different reinforcement learning algorithms are tried and contrasted to examine the convergence condition and performance parameters in the Blackjack domain.

B. Environment and State Space Design

Within the Blackjack environment, there is a systematic approach to represent a state that allows the main information needed to make the optimum decision within the Markovian frame. The state space is formulated so that it fulfills the Markov property and is still computationally tractable to reinforcement learning algorithms.

Basic State Space Formulation

The fundamental state representation is defined as a three-dimensional tuple:

$$S = (s_p, a_u, d_c)$$

where:

- $s_p \in \{4, 5, \ldots, 21\}$: Player hand sum — the total value of cards in the player's hand.
- $a_u \in \{0, 1\}$: Binary indicator for a usable ace — distinguishes between hard and soft hands.
- $d_c \in \{1, 2, \ldots, 10\}$: The dealer's visible upcard.

The possible values of the player hand, s_p, are the range [4, 21], all possible hand totals between the lowest combination (of 2 cards) and the highest property (a non-bust total). The indicators of the number of aces that can be used a_u, reflect the key strategic difference between having a soft hand (which is only one combination in which an ace in the hand has a value of 11) and a hard hand, because this differentiation affects the options of optimal play in a fundamental way. A single card value (10, J, Q, K) will be a group so that the dealer upcards d_c treats the aces and numbered cards in distinct categories since the cards are of strategic values in decision making.

$$\text{State Space Cardinality:} \quad |S| = 18 \times 2 \times 10 = 360$$

[4]

Enhanced State Space with Card Counting Integration

In cases where the game includes some analysis of card counting, the state space is enlarged to include information about the deck structure:

$$S = (s_p, a_u, d_c, c_b)$$

where:

- $c_b \in \{0, 1, 2, 3, 4\}$: True count bin index representing discretized deck composition.

The partition:$\{(-\infty, -5), \ [-5, 0), \ [0, 2), \ [2, 5), \ [5, \infty)\}$ defines five bins that categorize the full spectrum of deck favourability. This discretization reduces the infinitely expansive range of possible true counts into a finite and computationally manageable space, while preserving the essential strategic information necessary for betting and playing decisions.
Enhanced State Space Cardinality:

$$|S| = 18 \times 2 \times 10 \times 5 = 1{,}800$$

Hi-Lo Card Counting System

The Hi-Lo card counting system, developed by Harvey Dubner and refined by Edward Thorp, is incorporated into the enhanced state representation to capture deck composition dynamics. This system operates on the fundamental principle that Blackjack outcomes are determined by the probability distribution of drawing high-value cards (Ace, 10, J, Q, K) versus low-value cards (2–6) [4]. The Hi-Lo method makes decisions based on tracking the relative concentration of these card categories, as the depletion of low cards increases the probability of drawing favorable high cards, thereby shifting the expected value in favor of the player.

Running Count Mechanism: The Hi-Lo system has a running count (RC) which keeps track of the apparent lack of symmetry between the high and low cards with minor changes:

$$RC_{t+1} = RC_t + \text{count\_value}(card_t)$$

The counting system has a compensatory strategy so that, in case low cards are noted (which means that they should be taken off because they are of benefit to the player) the counter will be raised and vice versa when high cards are noticed (which signifies taking them off is of benefit to the dealer) the counter will be reduced:

$$\text{count\_value(card)} = \begin{cases} +1 & \text{if card} \in \{2, 3, 4, 5, 6\} \text{ (low)} \\ 0 & \text{if card} \in \{7, 8, 9\} \text{ (neutral)} \\ -1 & \text{if card} \in \{10, J, Q, K, A\} \text{ (high)} \end{cases}$$

This assignment scheme is a direct probabilistic representation of the effect that removal of cards has: positive assignments represent the enrichment of the deck in high cards (to the benefit of the player), negative assignments represent impoverishment of the high cards on the deck (to the benefit of the dealer). The mathematical principle that is being exploited in this case takes advantage of how the various cards affect the probabilities of Blackjack: Because high cards raise the possibility of the player blackjacks and dealer busts, as well as open up the possibility of rewarding double-down, the high cards bring greater advantages.

Dynamic Betting Strategy

The dynamic betting element raises the starting bet according to the count in play, trying to improve on the conditions of favourable probabilities:

$$\text{Bet}(TC) = \begin{cases} 1 & \text{if } TC < 2 \\ 2 & \text{if } 2 \leq TC < 4 \\ 4 & \text{if } TC \geq 4 \end{cases}$$

This development is the theorized interaction between true count values and the time of a player. The standard minimum bet maintains capital when the situation is not favourable at negative or low positive counts. As the true count goes higher, reflecting an increase in concentrations of good high cards, the size of bets also goes up accordingly to take advantage of the positive odds of winning a bigger hand.

Expected SARSA Algorithm

Expected SARSA is an off-policy temporal difference (TD) control algorithm that yields a more stable learning algorithm than conventional Q-learning and a superior sample efficiency element as compared to the on-policy SARSA [2]. The innovation is the major one since it involves computing the expected value over all possible next actions as opposed to computing that value based on a sampled action.
The Expected SARSA update rule is mathematically defined as:

$$\begin{aligned} Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \big[& R_{t+1} \\ & + \gamma \, \mathbb{E}_\pi \big[Q(S_{t+1}, A_{t+1}) \mid S_{t+1} \big] \\ & - Q(S_t, A_t) \big] \end{aligned} \tag{2}$$

[3]
Where the expected value term is calculated as:

$$\mathbb{E}_\pi \left[Q(S_{t+1}, A_{t+1}) \mid S_{t+1} \right] = \sum_a \pi(a \mid S_{t+1}) \, Q(S_{t+1}, a) \tag{3}$$

For an ε-greedy policy, this becomes:

$$\mathbb{E}\left[Q(S_{t+1}, A_{t+1})\right] = (1-\varepsilon)\max_a Q(S_{t+1}, a) + \frac{\varepsilon}{|A|}\sum_a Q(S_{t+1}, a) \tag{4}$$

This formulation reduces the variance of updates compared to Q-learning while allowing for off-policy learning [2], making it particularly suitable for the stochastic nature of Blackjack.

Monte Carlo Method

The Monte Carlo approach implements first-visit Monte Carlo control, learning directly from complete episode experiences, just the initial instance of every state (or state-action combination) in an episode gets used in value in the update procedure [2]. The method is particularly well-suited for Blackjack due to the episodic nature of the game and the fact that all rewards are received at episode termination.
The Monte Carlo update procedure follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{5}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}\left[G_t - Q(S_t, A_t)\right] \tag{6}$$

[3]
Where $N(S_t, A_t)$ represents the number of times state-action pair (S_t, A_t) has been visited. This can be equivalently written as:

$$Q(S_t, A_t) = \frac{1}{N(S_t, A_t)} \sum_{i=1}^{N(S_t,A_t)} G_t^{(i)} \tag{7}$$

The Monte Carlo method provides unbiased estimates of action values but typically requires more episodes for convergence compared to TD methods.

Enhanced Expected SARSA Variants

To improve convergence properties, we implemented enhanced variants incorporating:

- Adaptive Learning Rate Decay:
$$\alpha_t = \max(\alpha_{\text{min}}, \alpha_0 \times \delta_\alpha^t)$$
- Dynamic Exploration Scheduling:
$$\varepsilon_t = \max(\varepsilon_{\text{min}}, \varepsilon_0 \times \delta_\varepsilon^t)$$
- Optimistic Initial Values:
$$Q_0(s,a) = Q_{\text{opt}} > 0 \quad \text{for all } (s,a)$$

Such improvements promote exploration vigorously at the initial stage of training but guarantee convergence to steady plans.

C. Why Expected SARSA and Monte Carlo?

The paper draws the two fulfilling differentials in the field of reinforcement learning: Expected SARSA and Monte Carlo techniques. This has been a selective and strategic choice as it is believed to exemplify paradigms in RL: Expected SARSA is a part of the Temporal Difference (TD) learning subclass - learning algorithms that, to some degree, update value estimates as each step is completed, prior to episode termination. Expected SARSA particularly has the general advantage of being off-policy, with all the stability gains expected value calculations give over single action selections. Monte Carlo is an episode-based learning algorithm that waits to see the end of an episode before updating and learns using real returns, not bootstrapped estimates. This offers unbiased estimates and requires complete collections of the episodes.

III. Experiments and Evaluation

A. Implementation Architecture

The research infrastructure is implemented on a framework consisting of modules based on Python that allow control and reproducibility of the experiment:

1) Core Components:

- *BlackjackEnv Class:* Instantiates the place of the casino with regulations that can be configured.
- *RLAgent Base Class:* Supplies shared behavior (Q-table, epsilon greedy policy, evaluation modes).
- *Algorithm-Specific Agents:* MonteCarloAgent, Expected SARSA, Enhanced Expected SARSA
- Training Pipeline: The trained agent is the same across all checkpoints, with replicated evaluation processes.

2) Environment Standardization:

- *Deck Management:* 6-deck shoe and automatic reshuffle after fewer than 15 cards were left behind to provide similar game conditions in all experiments.
- *Card Counting Integration:* Hi-Lo counting system keeps a running count and calculates a true count.

3) Training Rule:

- *Episode Organization:* An episode should be one full hand of blackjack, including a deal and an end, with instant blackjack dealt with appropriately.
- *Action Space:* Binary choice (if 0 == stand, if 1 == hit) — kept deliberately simple to ensure concentration on more basic strategy learning, versus more advanced plays.
- *Learning Mode Control:* Agents switch between training and evaluation mode (epsilon greedy exploration and greedy exploitation, respectively) to guarantee an unbiased measurement of their performance.

All the algorithms undergo the same evaluation procedures as defined in Table I, for configurations of each of the algorithms. This blanket cross-testing is effectively able to give strong evidence of the choice of algorithm in different blackjack environments, as we can be sure that the performance difference is not merely based on environment-specific optimizations.

TABLE I: Environment Configuration Summary

Configuration	Description	Purpose
Basic Strategy	Standard six deck, dealer stands on soft 17	Baseline performance measurement
CPC System	Includes true count state information for betting	Advanced strategy evaluation
Resplit Aces	Allows re-splitting of ace pairs	Rule variation impact
Dealer Hole Card	Dealer's hole card is visible	Information advantage scenario

B. Results and Discussion

The performance of both reinforcement learning algorithms was assessed using four key metrics, each providing distinct insights into strategy effectiveness:

1) Win Rate: As follows: (wins / (wins + losses + draws)) This measure disregards pushes to concentrate on the quality of decision-making. With basic strategy, win percentages depend on the exact rules being played; they are usually in the 43-47% range owing to the fact that there are usually house advantages.

2) ROI - Return on Investment: total profit / total games ROI comprises the economic efficiency of the acquired strategy. Standard blackjack has a house advantage of 0.5-1%, creating a negative ROI of (-0.005 to -0.01) using basic strategy.

3) Loss Rate: Computed as losses/ (wins + losses + draws) Loss rates are the complement of win rates in order to have a full distribution outcome. Optimal play expected loss rates fall between 47-52%, with the optimal defensive strategy having a lower loss rate.

TABLE II: Performance Comparison of Algorithms Across Blackjack Variants

Algorithm	Strategy	Win Rate	ROI
ExpectedSARSA	Basic Strategy	41.76%	-0.1080
MonteCarlo	Basic Strategy	41.67%	-0.1331
ExpectedSARSA	CPC	41.89%	-0.0629
MonteCarlo	CPC	41.67%	-0.0757
ExpectedSARSA	Improved CPC	43.21%	-0.0607
MonteCarlo	Resplit of aces	41.60%	-0.0745
MonteCarlo	Dealer hole card	43.29%	-0.0423
ExpectedSARSA	Resplit of aces	40.69%	-0.0876
ExpectedSARSA	Dealer hole card	40.38%	-0.0941

The simulation results are as in Figure 1 reveal several key insights about the effectiveness of different reinforcement learning algorithms in blackjack strategy optimization. It presents a comprehensive overview of all nine experimental scenarios.

Basic Strategy Performance: In the basic strategy cases, the win rate of Expected SARSA was 41.76% and the ROI was -0.0629, and that of Monte Carlo was 41.35% and the ROI was -0.0757. This 0.41 percentage point lead of Expected SARSA corresponds to about 41 extra wins in 10,000 games, which turns into significant monetary differences during an extended play. This is because Expected SARSA performance seems to be more stable because it takes an expected value of future moves, unlike Monte Carlo, which uses single-episode returns.

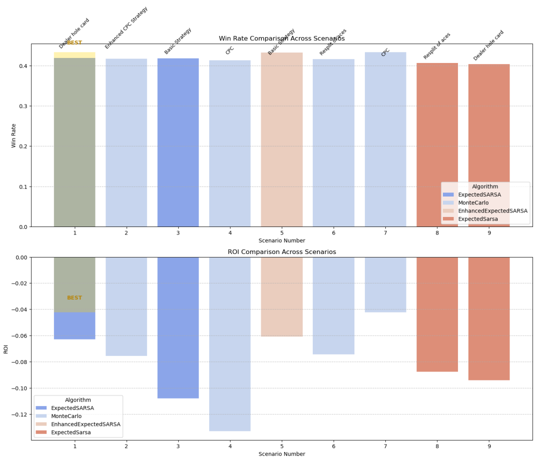

Fig. 1: Performance Analysis

The fact that all the cases offer negative ROI values is an additional confirmation of the generally accepted mathematical fact that blackjack is a game with an in-built advantage on the side of the house. Nevertheless, the extent of such losses greatly differs depending upon both the algorithm decision and the strategic development, with Expected SARSA usually reducing the losses better.

Card Counting Integration (CPC Strategy): Interesting differences in the algorithm were seen with the implementation of card counting using the CPC (Card Point Counting) system. Expected SARSA with CPC performed with a 41.89% win rate and a higher level of loss in ROI at -0.1080 in comparison to Monte Carlo with CPC, which performed with 41.67% wins with -0.1331 loss in ROI. The decreasing returns on their initial investment show that the agents had learned how to raise the amount of their bet in the positive count compared to what it had been before, but they might have become excessively aggressive, experiencing greater loss when their approach failed. This paradox demonstrates a crucial weakness of the current RL applications to blackjack: the algorithms might acquire the ability to make the best hit/stand choices, but fail to make the tricky requests connected to the management of the risks with regard to making effective bets. The integration of card counting led to a win rate increase of around 0.04-0.32 percentage points, whereas the associated returns on investment losses indicate the need for more advanced reward shaping or multi-objective optimization.

Enhanced Algorithm Performance: The Enhanced Expected SARSA variant reaped the best potential with a 43.21 percent win rate, passing the highest ROI of -0.0607 as seen in Figure 1. This is quite encouraging in comparison with a normal algorithm, and the level of winning stands at more than 43 percent, which is scarcely observed in blackjack simulations. This 1.45 percentage point gain comes down to about 145 more wins per 10,000 games over standard Expected SARSA, which shows the importance of algorithmic refinements to complex sequential decision problems.

Dealer Hole Card Visibility: Highly varied results noted

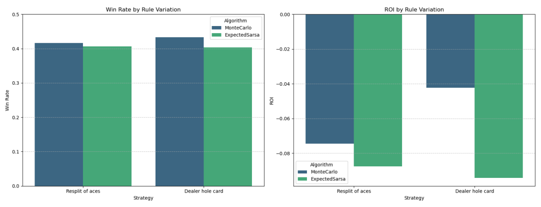

Fig. 2: Rule Variations Performance

in Figure 2 among algorithms were seen in the cases of visible dealer hole cards (Scenarios 7 and 9). Monte Carlo also performed well, making 43.29 win percent with -0.0423 ROI, which is the second-best overall performance in any situation. This implies that the Monte Carlo episode-based learning is quite optimal whenever there is a utilization of perfect information cases. On the other hand, on the right panel of Figure 1, Expected SARSA with visible hole cards performed at a much lower level (40.38% win rate, -0.0941 ROI), meaning that its location in the time learning strategy might not be able to utilize the full informational advantage. The counterintuitive outcome implies that various RL algorithms are potentially inefficient at exploiting various kinds of strategic advantages.

Resplit Aces Rules: The resplitting of the aces scenarios (Scenarios 6 and 8) on the left panel of Figure 2 showed more modest improvements. Monte Carlo won 41.60 percent and had -0.0745 ROI, whereas Expected SARSA was able to win only 40.69% percent and had -0.0876 ROI. These fairly low gains indicate that resplitting aces is mathematically favorable, but the corresponding variation in the rules was not efficiently used by both algorithms, perhaps because the occurrence of a pair of aces is low and because the rule further complicates decision trees.

C. Strategic Insights from Policy Analysis

An essential understanding of the learning process to play blackjack based on every algorithm can be drawn with the help of the policy visualization as in Figure 3. In the case of both algorithms, they were able to learn foundational principles of the basic strategy, including:

- Conservative play with hard totals: Both algorithms were taught to stand on hard 17-21 irrespective of whether the dealer was showing or not.
- Low totals aggressive play.
- Dealer dependent calls: Optimisation on strategy subject to dealer upcard

Nonetheless, slight differences in policies learnt are the reason why there is a difference in performance. The expected SARSA proved more consistent in its domain across similar situations.

The training outcomes in Figure 1show that Expected SARSA typically needed fewer episodes to settle down on a steady performance than Monte Carlo. The difference in efficiency is because Expected SARSA gives more chances to update the Q-values after solving every action, and hence, it is more efficient to learn. Monte Carlo, which needs full

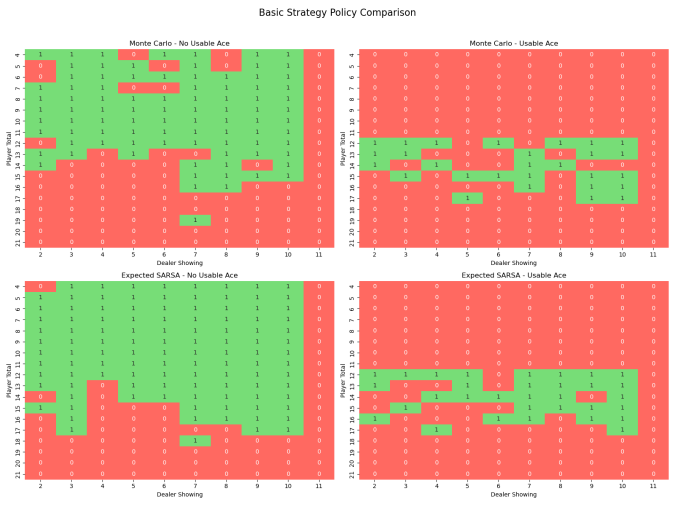

Fig. 3: Strategy Comparison

episodes to accomplish updates, converged more slowly, but in certain cases attained a better final performance.

IV. Conclusion

This study deployed and assessed various reinforcement learning methods to optimize blackjack strategy, compared Expected SARSA with Monte Carlo with enhanced Expected SARSA and Monte Carlo in different game settings, such as basic strategy, card counting (CPC), and changed rules. The experimental results have shown that all of the reinforcement learning agents had optimally learned patterns in playing blackjack, with the Enhanced expected SARSA agent performing the best in terms of a combination with the card counting techniques. Further directions of research are the exploration of deep reinforcement learning strategies to more complex games of blackjack, competitive multi-player game environments, testing learned policies, and the equivalence to real casino data in real-world settings, including the consideration of complex bankroll management with risk-adjusted measures of performance. Casino games, meetings, and reinforcement learning offer a fertile ground to simultaneously improve both theoretical knowledge of RL algorithms and practical implementation aspects in making decisions under uncertainty.

References

[1] N. A. Bond, “Basic strategy and expectation in casino Blackjack,” *Organizational Behavior and Human Performance*, vol. 12, no. 3, pp. 413–428, Dec. 1974, publisher: Elsevier BV. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0030507374900610

[2] E. O. Thorp, *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. New York: Vintage Books, 1966.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

[4] Z. Liu and G. Spil, “Learning Explainable Policy For Playing Blackjack Using Deep Reinforcement Learning (Reinforcement Learning).”