Improving BlackJack Strategies Using Reinforcement Learning

Ali Asghar
Master in Artificial Intelligence
University of Applied Sciences Wurzburg-Schweinfurt
ali.asghar@study.thws.de

***Abstract*—This research article provides an in-depth investigation of Blackjack game techniques using simulated Environments. Blackjack, a popular card game in casinos across the world, requires players to have a hand total that is closer to 21 than the dealer's but not higher. Key methods include deciding whether to hit, stand, double down, or divide pairs in order to increase your chances of winning against the dealer. The research focuses on two main environments: a basic blackjack environment and a complete card counting environment. The Basic Blackjack Environment includes fundamental gaming features including card drawing and betting, as well as player actions like hitting, standing, doubling down, and splitting. The Complete Card Counting Environment expands on this paradigm by including sophisticated card counting strategies, notably the High-Low strategy, that dynamically change betting and playing decisions depending on the real count. SARSA (State-Action-Reward-State-Action), Monte Carlo On-policy, Monte Carlo Off-policy, and Temporal Difference (TD) agents were deployed in both environments to evaluate the effectiveness of different methods. The performance of these agents was examined using standard metrics to measure their efficiency in learning optimal decision-making techniques. Studies showed that using card counting techniques significantly improve agent performance, resulting in better win rates and average winnings in comparison to basic strategy. In addition, modifying bets and actions based on the correct count generally enhances profitability, highlighting the practical benefits of card counting in Blackjack. The aim of this research is to increases the understanding of game theory and artificial intelligence by implementing and analyzing advanced Blackjack strategies in simulated environments. The findings provide beneficial insights for both academic study and advertising casino gaming applications.**

***Index Terms*—BlackJack, Reinforcement Learning Agents, Machine Learning, Artificial Intelligence, Monte Carlo, SARSA**

I. Introduction

Blackjack, is one of the most popular and commonly played casino card games. The main objective of the game is for players to get a hand worth closer to 21 than the dealer's without exceeding it. Since the beginning, many tactics have been devised to maximize player decisions, including as when to hit, stand, double down, or divide pairs. These techniques try to enhance the player's odds of winning while minimizing the casino's edge.

Blackjack has been widely explored in the domains of game theory and artificial intelligence (AI). Early research, such as Thorp's (1966), offered card counting strategies that dramatically increase player advantage by measuring the ratio of high to low cards left in the deck. Thorp's groundbreaking work on the High-Low card counting method, documented in his landmark book Beat the Dealer (Thorp, 1966), transformed the understanding of Blackjack. His approaches revealed that the casino's edge could be overcome by methodical play and strategic modifications, setting the framework for future study into both theoretical and practical elements of the game.[1]

Thorp's foundation has been built upon in subsequent research, which has looked into numerous aspects of Blackjack strategy. Griffin's Theory of Blackjack (1979) provides a detailed review of statistical models and improved card counting strategies[2]. Epstein's The Theory of Gambling and Statistical Logic (1977) provided a larger context for gambling tactics within statistical theory, leading to a better understanding of the probabilistic features of Blackjack.[3] Reinforcement learning (RL) has recently emerged as a useful method for designing and improving tactics in a variety of contexts, including games. Sutton and Barto (2018) thoroughly researched RL approaches such as SARSA, Monte Carlo methods, and Temporal Difference (TD) learning, proving their effectiveness in learning optimum policies through interaction with the environment [4]. These techniques have been effectively used to a variety of games, including board games such as Go and video games like Atari.

Building on this foundation, My study investigates the application of RL approaches to blackjack. I created and test two main environments: a basic blackjack environment and a full card counting environment. The Basic Blackjack Environment recreates conventional gameplay features such as card drawing, betting, and player actions such as hit, stand, double down, and split. In contrast, the Complete Card Counting Environment combines sophisticated card counting approaches, notably the High-Low strategy, which allows for dynamic modifications in betting and playing decisions depending on the real count. [1][2]

In this study, I introduce a novel way for enhancing the Blackjack strategy through the use of reinforcement learning techniques. My objective is to create a Blackjack player who performs better than traditional methods, both with regard to the Basic strategy and an extensive Card Counting method derived from Thorp's Beat the Dealer [1]. To do this, I implement a number of RL agents in both contexts. These include SARSA (State-Action-Reward-State-Action), Monte Carlo On-Policy, Monte Carlo Off-Policy, and Temporal Difference (TD) agents. Each agent is programmed to learn and adjust its strategy

based on interactions in the game environment, using reinforcement learning concepts to improve decision-making processes [4].

The use of Q-Learning to Blackjack demonstrates how RL approaches can improve traditional strategies by adjusting to the game's dynamic nature [5]. Similarly, studies investigated multiple RL ways to deal with the imperfect information inherent in Blackjack, demonstrating the adaptability of these methods in improving decision-making under uncertainty [6].

This study intends to add to game theory and AI by implementing and analyzing advanced Blackjack tactics in simulated environments. Our findings not only support the practical benefits of card counting in Blackjack, but also provide useful insights for academic study and practical casino gaming applications. We use reinforcement learning to push the boundaries of strategic play in Blackjack, offering new perspectives on how AI might modify classic gaming methods.

This research paper has the follwong structure and subsections: Section 2 (Methodology part of the paper where I explain different deployed Reinforcement Learning agents and their way of working to improve the BlackJack strategy) Section 3 (Experiments and Evaluation results,describes my findings and their analysis through the proposed Methodology) Section 4(Conclusion)

II. Methodology

The purpose of blackjack is to create an accurate strategy for decision-making amid uncertainties and limited information. while playing.the major challenge is accurately estimating the values of various state-action pairs to optimize expected returns over time for participants. SARSA(State-Action-Reward-State-Action),TD(Temporal Difference) ,Monte Carlo on Policy and off Policy techniques are employed specifically for the blackjack game to address this issue. These strategies address the game's inherent variations and uncertainties, resulting in a framework for precisely forecasting the best course of action in various blackjack scenarios.

A. SARSA (State-Action-Reward-State-Action)

The SARSA (State-Action-Reward-State-Action) agent is a reinforcement learning system that uses the on-policy temporal difference (TD). It modifies the action-value function based on the agent's interactions with the environment and the policy it is learning. The basic idea behind SARSA is to calculate the value of performing a specific action in a specific state while conforming to current policy. This agent learns from the actions it performs and updates its knowledge using the following update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \quad (1)$$

In this equation:

- $Q(s_t, a_t)$ represents the estimated value of taking action a in state s.
- α is the learning rate, which controls how much new information overrides the old information.
- γ is the discount factor, determining the importance of future rewards.
- r_{t+1} is the reward received after transitioning to the next state.

The epsilon-greedy approach is used by the SARSA agent to balance between exploration and exploitation. This technique allows the agent to explore the action space by selecting random actions with a probability of , but largely relying on the learnt action-value function for optimal judgments. [4][5].

B. TD (Temoral difference) Agent

The Temporal Difference (TD) learning agent is a fundamental reinforcement learning technique that combines Monte Carlo and dynamic programming principles. Unlike Monte Carlo approaches, which require waiting until the conclusion of an episode to update value estimates, TD learning updates value estimates at each time step based on the observed reward and the estimated value of the next state. This enables TD agents to learn more efficiently and effectively in contexts that involve ongoing interactions. The TD learning update rule is as follows:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right] \quad (2)$$

Here:

- $V(s_t)$ is the estimated value of the current state s_t.
- r_{t+1} is the reward received after transitioning to the next state.
- α is the learning rate.
- γ is the discount factor, controlling the weight of future rewards.

TD learning is particularly useful in Markov Decision Processes (MDPs) because it can build or update estimates based on other learnt estimates without waiting for final outcomes.

C. Monte Carlo Techniques

Monte Carlo methods are a core collection of strategies in reinforcement learning (RL) that use the notion of learning from episodes of experience rather than individual transitions. These methods are especially valuable when an environment model is unavailable since they allow for the estimation of value functions based solely on the agent's own experience.

In essence, Monte Carlo methods calculate the worth of states and actions by averaging the returns (cumulative rewards) observed after visiting or performing such actions. This methodology differs from Temporal Difference (TD) learning approaches, which update value estimations incrementally at each time step.

The Monte Carlo technique assesses and optimizes policies based on the average return from repeated occurrences. An episode is a collection of states, actions, and rewards that culminates in a terminal state. Monte Carlo methods are model-free, which means they don't require knowledge of the dynamics of the environment, just the ability to generate episodes through interaction with it.

Monte Carlo methods rely heavily on the law of large numbers: as the number of episodes rises, the average return

approaches the expected return. This makes Monte Carlo approaches appropriate for situations in which episodes are finite and can be simulated several times.

1) Monte Carlo On-Policy: Monte Carlo On-Policy approaches focus on learning the value of the policy that the agent is presently implementing. This means that the same policy is utilized to create behavior (i.e., actions) as well as to assess and improve the policy. On-policy approaches try to estimate the action-value function q (s,a), which represents the expected return of taking action a in state s and following policy .

The update rule for the Monte Carlo On-Policy agent is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[G_t - Q(s_t, a_t)\right] \tag{3}$$

where

- G_t is the total return following time t.
- α is the learning rate.
- $Q(s_t, a_t)$ represents the action-value function.

Monte Carlo On-Policy agents use the epsilon-greedy approach to ensure enough exploration. This helps the agent learn accurate estimations of the action values and improves its decision-making overtime.

2) Monte Carlo Off-Policy: Monte Carlo Off-policy methods use the concept of importance sampling to determine the best policy irrespective of the policy being followed. Off-policy learning enables the agent to analyze and improve the target policy by utilizing data from a distinct behavioral policy. This is especially beneficial when the target policy is deterministic or greedy while the behavior policy is exploratory. The target policy is the one being studied, whereas the behavior policy is employed to generate episodes and explore the environment. Importance sampling ratios are used throughout the off-policy process to account for the discrepancy between the behavior policy and the target policy. The update rule for the MC Off-Policy agent is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{\alpha}{C(s_t, a_t)} \left[G_t - Q(s_t, a_t)\right] \tag{4}$$

where:

- $C(s_t, a_t)$ is the cumulative sum of importance sampling ratios.
- G_t is the return following state s_t.
- α is the learning rate.

Monte Carlo Off-Policy approaches ensure that the agent can learn optimal policies while following to a distinct policy, which gives flexibility in learning from different scenarios. This approach is powerful because it allow the agent to use prior knowledge, regardless of those acquired under different policies, to better their current decision-making.

D. Environment and their Agents Iterations

Blackjack strategy relies on both basic strategy and a counting mechanism. These aspects play an important influence in the agent's gameplay decision-making tries to maximize the agent's performance and increase its chances of winning.

1) Basic Blackjack Environment: The Basic Blackjack Environment is an interactive environment that follows standard blackjack rules and serves as a starting point for testing reinforcement learning (RL) techniques. This environment utilizes a set of established rules developed via intensive statistical analysis, allowing the agent to make the most accurate decisions based on the circumstances at hand. In this Environment, the game employs a conventional deck of 52 cards, which is shuffled before each round to assure randomization. At the start of each round, the player and the dealer are each given two cards. The player's cards are visible, whereas the dealer has one card face up (the upcard) and one card face down (the hole card). The player can take one of four actions: hit (requesting additional cards to increase the total value of the hand), stand (retaining the current hand and ending the turn), double down (doubling the initial bet in exchange for exactly one more card), or split. According to traditional casino regulations, the dealer must hit until their hand totals 17 or higher, even a soft 17 (a hand totaling 17 with an Ace scored as 11). The goal is to get a hand value closer to 21 than the dealer but not surpassing 21. Winning circumstances include the player's hand being closer to 21 than the dealer's, the dealer busting by going over 21, or attaining a natural blackjack (an Ace and a 10-value card) unless the dealer also has a blackjack, in which case the game is a push.

- Agent Interactions:

In this Environment, the RL agents are constantly learning and adapting. They begin with no past information and participate in exploration to better comprehend the game mechanics. As they play more games, they improve their strategy depending on the rewards and results of their activities. The agents learn to analyze the probabilities of various events and make intelligent decisions in order to increase their chances of winning. Through repeated play, kids enhance their decision-making processes, harnessing their expanding knowledge to dynamically alter their strategies.

2) Complete Card Counting Environment: The Complete Card Counting Environment expands on the Basic Blackjack Environment by including advanced card counting tactics, particularly the High-Low counting strategy. This environment replicates a more complicated and realistic casino setting, with the user making dynamic betting and playing decisions based on the number of cards remaining in the deck. In this Environment, the High-Low counting method provides values to cards: high cards (10s, face cards, and Aces) are tallied as -1, low cards (2-6) as +1, and neutral cards (7-9) as 0. By keeping track of the cards dealt, the player can estimate the percentage of high to low cards remaining in the deck. A positive count implies a greater number of high cards, which favors the player, and a negative count shows that more low cards remain, which favors the dealer. The player's betting strategy changes based on the count. When the count is positive, the agent raises its wager to take advantage of the favorable deck structure. In contrast, when the count is negative, the agent minimizes its stake in order to limit losses.

- Agent Interactions:

In this Environment, RL agents process and respond to increasingly complicated information. They constantly learn and change their strategy by keeping track of the cards given. Agents dynamically alter their betting and playing methods according on the count, striking a balance between urgent hand decisions and long-term count-based strategies. They change their strategy, such as standing on lesser totals or placing insurance bets, to take advantage of the dealer's heightened chance of busting. Agents' performance improves dramatically over time as they learn and adapt to the count, resulting in higher win rates and average winnings than with the basic technique alone.

III. Experiments and results

In this part, I review the experiments carried out to evaluate the performance of several reinforcement learning agents—SARSA (on-policy), TD(0) (off-policy), Monte Carlo (on-policy), and Monte Carlo (off-policy)—in the context of improving Blackjack techniques. I evaluate each agent's win rates and average win rates over different discount factors () to evaluate their efficiency and performance.

A. Basic Strategy

1) Experimental Setup: The studies has been carried out in the previously specified Basic Strategy Blackjack environment. To achieve a balance between exploration and exploitation, each agent has been trained across 100,000 episodes using an epsilon-greedy strategy (= 0.05). Performance indicators (win rate and average wins) were recorded for discount factors = 0.7, 0.8, 0.9, and 0.99. These metrics give information on the agent's capacity to learn and implement effective Blackjack strategy.

2) Results For Basic Strategy: Tables 1 and 2 present a summary of the results for each agent. These tables provide the win rate and average wins for each agent at various discount factors.

2*Gamma	Basic Strategy			
	SARSA(on-policy)		TD(0)(off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	37%	9.93	41%	35.61
0.8	39%	24.42	40%	24.73
0.9	37%	18.57	38%	16.84
0.99	39%	24.96	39%	19.09

TABLE I
SARSA (ON-POLICY) AND TD(0) (OFF-POLICY) BASIC STRATEGY PERFORMANCE

2*Gamma	Basic Strategy			
	Monte Carlo (on-policy)		Monte Carlo (off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	40%	26.45	39%	22.11
0.8	36%	12.89	40%	21.29
0.9	38%	16.22	41%	27.84
0.99	38%	10.55	39%	15.78

TABLE II
TABLE 2: MONTE CARLO (ON-POLICY) AND MONTE CARLO (OFF-POLICY) BASIC STRATEGY PERFORMANCE

All these experiments are done with the gamma 0.7,0.8,0.9,0.99 and with the same epsilon value 0.1 and alpha value 0.5

3) Analysis:

a) SARSA (on-policy) vs. TD(0) (off-policy): The SARSA agent, being an on-policy approach, closely adheres to the policy being learned. The results indicate a constant win rate of 37 per to 39 per across various discount factors, with the greatest average win rate at y = 0.99. This demonstrates that SARSA efficiently balances immediate and future incentives, resulting in consistent performance.

The TD(0) agent, an off-policy technique, has a greater win rate compared to SARSA, reaching 41 per when y = 0.7. The average win rates for TD(0) were much higher, particularly at smaller discount factors, suggesting that it may be more efficient in learning optimum strategies by using off-policy changes. This is consistent with the prevailing belief that off-policy algorithms can learn more efficiently by employing diverse exploration strategies [4].

b) Monte Carlo (on-policy) vs. Monte Carlo (off-policy): Monte Carlo approaches, which update value estimations using full episodes, performed differently depending on the policy type. The Monte Carlo agent performed well at y = 0.7, with a 40 per win rate. However, its performance declined marginally at larger discount factors. This tendency implies that, while Monte Carlo on-policy can perform well over shorter time horizons, it may struggle with longer-term planning due to its reliance on episodic returns.

The Monte Carlo off-policy agent was more adaptable, with a peak victory rate of 41 per at y = 0.9 and the greatest average win rate of 27.84. This agent's off-policy nature allows it to employ importance sampling to better assess the value of the target policy, resulting in more robust learning results even when trained under different behavior policies.

B. Complete Card Count System

In this part, I present the experiments carried out to assess the performance of several reinforcement learning agents—SARSA (on-policy), TD(0) (off-policy), Monte Carlo (on-policy), and Monte Carlo (off-policy)—in the context of a Complete card counting system for blackjack. I analyze each agent's win rates and average win rates under various discount factors to evaluate their efficacy and performance.

1) Experimental Setup: The studies has been carried out in the previously established Complete Card Coutning Blackjack environment. To achieve a balance between exploration and exploitation, each agent was trained across 100,000 episodes using an epsilon-greedy strategy (= 0.05). Performance measures, such as win rate and average wins, were measured for discount factors y = 0.7, 0.8, 0.9, and 0.99. These metrics give information on the agent's capacity to learn and implement effective Blackjack strategy.

2) Results for Complete Card Counting System: Tables III and IV present a summary of the results for each agent. These tables provide the win rate per and average wins for each agent at various discount factors.

2*Gamma	Complete Card Counting System			
	SARSA (on-policy)		TD (0) (off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	38%	10.15	39%	22.11
0.8	36%	9.55	37%	10.32
0.9	38%	12.42	40%	26.29
0.99	37%	7.88	36%	1.27

TABLE III

TABLE 3: SARSA (ON-POLICY) AND TD (OFF-POLICY) COMPLETE CARD COUNT PERFORMANCE

2*Gamma	Complete Card Counting System			
	Monte-Carlo (on-policy)		Monte-Carlo (off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	40%	18.20	40%	13.40
0.8	40%	21.24	38%	14.18
0.9	38%	10.03	39%	14.47
0.99	36%	1.10	40%	19.14

TABLE IV

TABLE 3: MONTE CARLO (ON-POLICY) AND MONTE CARLO (OFF-POLICY) COMPLETE CARD COUNT PERFORMANCE

3) Analysis:

a) SARSA (on-policy) vs. TD(0) (off-policy): The SARSA agent achieved a victory rate of 36 per to 38 per throughout the Complete card counting system, with the highest average win rate at y = 0.9. The findings indicate that, while SARSA maintains consistent win rates, it struggles to optimize wins in the Complete card count scenario.

In contrast, the TD(0) agent outperformed with win rates up to 40 per at y = 0.9 and much better average win rates at lower discount factors. The maximal average win rate of 26.29 at y = 0.9 demonstrates TD(0)'s capacity to use off-policy updates for successful strategy learning. This advantage correlates with the recognized benefits of off-policy approaches in dealing with diverse exploration strategies and more robust policy modifications.

b) Monte Carlo (on-policy) vs. Monte Carlo (off-policy): In the Complete card counting system, the Monte Carlo on-policy agent continuously obtained a win rate of 40 for y = 0.7 and 0.8, with a significant average win rate of 21.24 at y = 0.8. These findings show that Monte Carlo on-policy may efficiently use episodic returns to optimize strategies in a card counting environment.

While the Monte Carlo off-policy agent had significant win rates, its average winnings varied. It has a peak victory rate of 40 per at y = 0.7 and 0.99, with the greatest average win rate of 19.14 at y = 0.99. The results imply that off-policy updates allow this agent to efficiently modify its approach based on the total card count, thereby boosting performance over time.

C. Rewards per Episode:

In this section I presents the reward per episode plots for each algorithm which give further information about the performance dynamics and variability across episodes for each agent and for both Environments.These plots can provide the useful insights in agents regarding the decision-making techniques.

1) Basic Strategy :

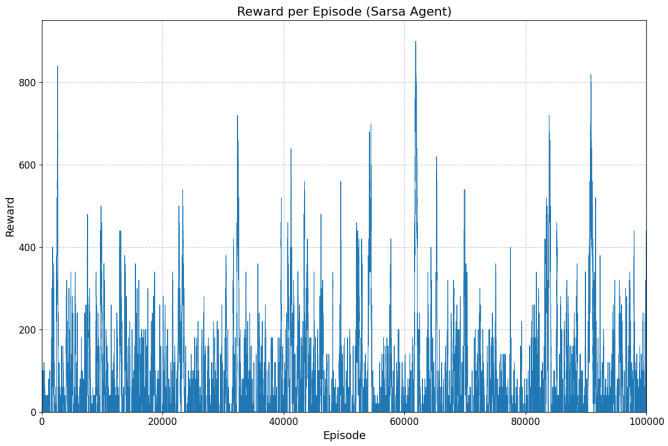

Fig. 1. SARSA (on-policy) Basic Strategy

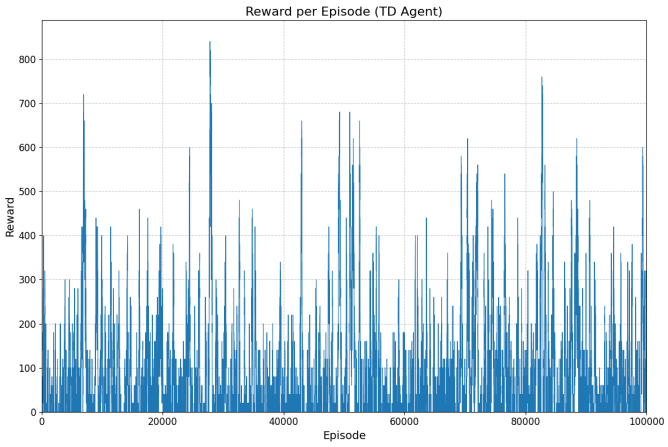

Fig. 2. TD(0) off-policy Basic Strategy

a) SARSA (On-Policy) Reward per Episode Plot: Based on the Fig.1. The SARSA agent's reward patterns are very random, ranging from 0 to about 900 per episode. The reward distribution is defined by frequent low-magnitude rewards and infrequent high-magnitude rewards. This pattern is stable throughout 100,000 instances, indicating that there is no substantial long-term learning trend. The considerable variability and rare peaks demonstrate that while SARSA can obtain significant rewards in some episodes, its overall performance remains uneven, indicating its sensitivity to the exploration-exploitation balance.

b) Temporal-Difference (TD) Agent: Fig.2 shows that the TD agent shows a more consistent reward pattern, with benefits ranging from 0 to 800. The moderate variability and higher frequency of moderate rewards indicate that the TD agent strikes a balance between exploration and exploitation. Although high-reward peaks persist, they are less common than in the Monte Carlo Off-Policy and SARSA plots.This suggests that the TD agent is learning from its experiences and gradually improving its methods, resulting in more consistent performance over time.

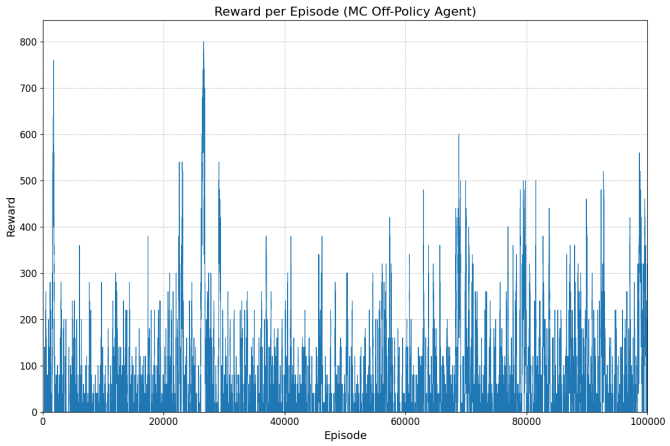

Fig. 3. Monte Carlo off-policy Basic Strategy

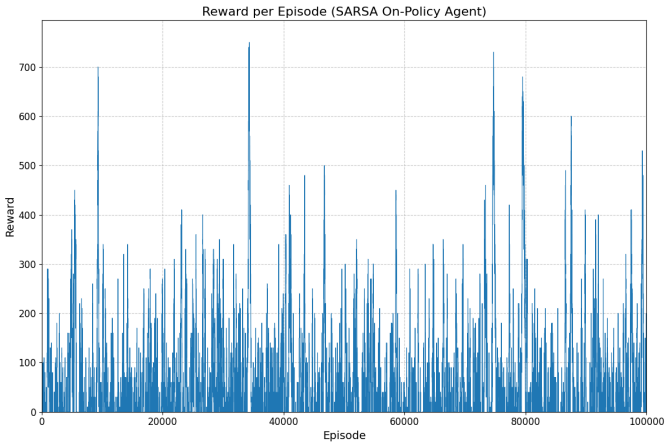

Fig. 5. SARSA(on-policy) Complete Card Count

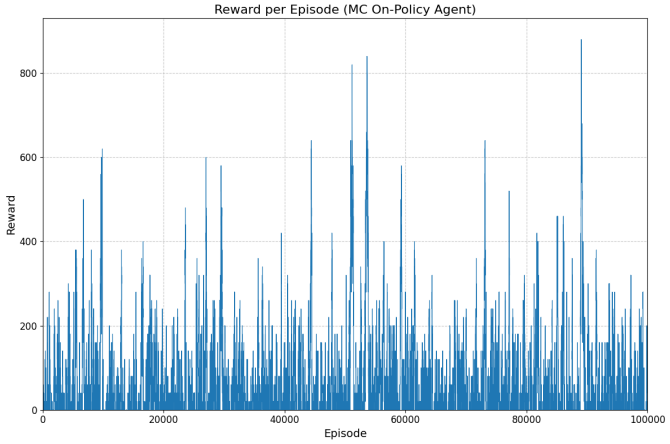

Fig. 4. Monte Carlo on-policy Basic Strategy

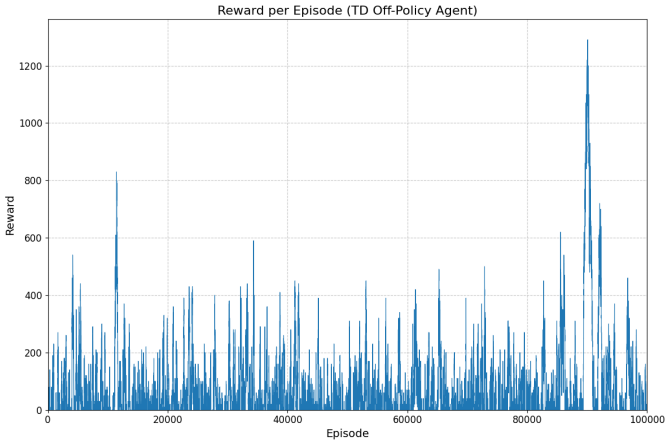

Fig. 6. TD agent Complete Card Count

c) Mone-Carlo off-policy: Fig. 3 shows that the Monte Carlo Off-Policy agent's plot displays a significant degree of unpredictability, with rewards ranging from 0 to 800. The lack of a distinct rising trend across 100,000 episodes suggests that the agent does not consistently learn from its failures in order to increase performance over time. The infrequent high rewards indicate that the agent occasionally discovers excellent policies but fails to keep them.

d) Monte Carlo on-policy: : Fig. 4. shows that the Monte Carlo On-Policy agent plot demonstrates a great degree of unpredictability, with rewards ranging from 0 to over 800. The benefits are intermittent and fluctuate significantly, like with the Monte Carlo Off-Policy agent. There is no apparent pattern demonstrating continuous progress over time, implying that the Monte Carlo On-Policy agent, like its off-policy counterpart, fails to sustain successful policies on a consistent basis.

2) Complete Card Count System:

a) SARSA (on-policy): In the Complete Card Counting context, the SARSA On-Policy agent performs consistently but with limited efficacy. The Fig. 5 demonstrates that rewards typically range between 0 to 400, with rare surges up to 700. This pattern remains very consistent throughout 100,000 occurrences, indicating that the agent soon finds a performance.

The agent has a significant short-term variability, which reflects the stochastic nature of blackjack. However, it fails to demonstrate meaningful long-term improvement. The continuous distribution of rewards throughout the trial indicates that the SARSA agent is not learning from its mistakes or perfecting its tactics over time. While it performs consistently, it is unable to leverage on the potential benefits of card counting to significantly boost its reward.

b) Temporal-Differenc TD Agent (Off-Policy): The Fig. 6 demonstrate that the agent performs dynamically well, with rewards ranging from 0-600, and then occasionally exceeding 1300. After 80,000 episodes, performance increases, showing that learning and strategy are being refined. A late 1300-point surge implies the possibility of extraordinary plays. The agent displays adaptive learning and steady progress over time.

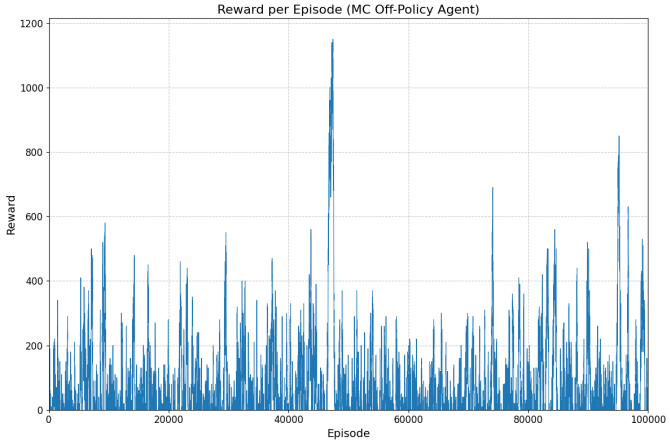

Fig. 7. Monte Carlo off Policy Complete Card Count

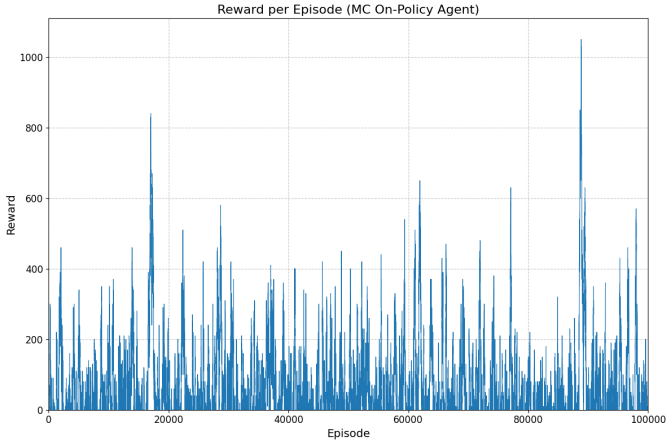

Fig. 8. Monte Carlo on Policy Complete Card Count

c) Monte Carlo Off-Policy: The Fig. 7. shows that the Complete card counting environment, the MC off-policy agent follows a similar high-variance pattern, with rewards occasionally reaching 1200. The distribution remains erratic, with frequent low awards and infrequent big rewards. This shows that, while the agent can take advantage of advantageous situations in certain episodes, its overall learning and performance remain unstable over the course of 100,000 episodes.

d) Monte Carlo On-Policy: The Fig. 8. shows that in the Complete card counting environment, the MC on-policy agent follows a similar pattern as the MC off-policy agent, with incentives ranging from 100 to 1000. Rewards remain very variable, reflecting inconsistent performance. There are regular oscillations and peaks, but no discernible long-term learning trend is visible.

IV. Conclusion

In this study, I conducted an in-depth comparative analysis of four reinforcement learning algorithms—SARSA (On-Policy), Temporal Difference (TD), Monte Carlo (MC) On-Policy, and Monte Carlo (MC) Off-Policy—in two different Blackjack environments: Basic Blackjack Strategies and Complete Card Counting. The goal was to evaluate these algorithms' performance, learning dynamics, and adaptability throughout 100,000 episodes in each Environment.

The experiments revealed significant patterns how each algorithm learned and adapted to its environments. The Temporal Difference (TD) learning system regularly outperformed the others, with rewards increasing steadily throughout both environments. This demonstrates TD's strong potential for learning and strategy optimization, which is critical for applications that require constant performance over time. The SARSA (On-Policy) algorithm exhibited great reward variability, achieving major rewards in select episodes but lacking consistency. Similarly, both Monte Carlo approaches (On-Policy and Off-Policy) demonstrated considerable moves in reward patterns, with occasional high-reward actions but generally inconsistent performance. These patterns indicate that, while these algorithms may learn efficient tactics, their performance is inconsistent and less dependable than TD.

My experimental results, which are presented in precisely reward-per-episode graphs and tables, give further information about each algorithm's learning processes and efficiency. The reward-per-episode plots clearly show the variability and trends in the performance of each algorithm over time. These visualizations, when combined with the tabulated statistics, highlight both the benefits and drawbacks of each approach in adapting to the two Blackjack environments. Overall, my results emphasize the significance of stability and consistency in reinforcement learning systems for improving the BlackJack strategies. The TD algorithm's exceptional performance highlights its promise for practical applications that require ongoing development and dependable strategy adaption. Future study might focus on improving Monte Carlo technique consistency and investigating the stability elements that contribute to the TD algorithm's effectiveness. This work gives important insights into the optimization of reinforcement learning approaches, opening the door for more successful implementations in complicated, real-world circumstances.

References

[1] E. O. Thorp, Beat the Dealer, New York, 1966.
[2] Griffin, P. A. (1979). Theory of Blackjack. Huntington Press.
[3] Epstein, R. A. (1977). The Theory of Gambling and Statistical Logic. Academic Press.
[4] Sutton, R. S., Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press..
[5] de Granville, C. Applying Reinforcement Learning to Blackjack Using Q-Learning.
[6] Wilson, A. A. Blackjack: Reinforcement Learning Approaches to an Incomplete Information Game.