Optimizing 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. This 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 has been 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 Complete 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. My 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. I 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:

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

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 (Temporal 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:

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

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 Interaction

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 Interaction:

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, like 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 Interaction:

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.

Gamma	Basic Strategy			
	SARSA(on-policy)		TD(0)(off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	39%	17.18	41%	29.93
0.8	42%	31.25	41%	25.15
0.9	41%	24.56	39%	22.33
0.99	39%	23.14	40%	18.16

TABLE I
TABLE 1 : SARSA (ON-POLICY) AND TD(0) (OFF-POLICY) BASIC STRATEGY PERFORMANCE

Gamma	Basic Strategy			
	Monte Carlo (on-policy)		Monte Carlo (off-policy)	
	Win Rate %	Avg Win Rate	Win Rate %	Avg Win Rate
0.7	40%	25.53	39%	18.71
0.8	41%	23.77	40%	19.37
0.9	39%	18.91	38%	13.17
0.99	38%	15.33	39%	13.86

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's on-policy approach leads to win rates ranging from 39 to 42 per, with the highest average win rate of y=0.8. This indicates SARSA's ability to balance present and future incentives successfully, peaking at a 42 per win rate and an average win rate of 31.25 when modestly prioritising future rewards.

The TD(0) agent outperforms SARSA as an off-policy method, with a peak win rate of 41 per at y = 0.7. when smaller discount factors, the average win rate is substantially greater, reaching 29.93 when y= 0.7. This shows that TD(0) is more effective in learning optimum strategies through off-policy adjustments, which is consistent with the idea that off-policy algorithms learn more efficiently by applying different exploration techniques [4].

b) Monte Carlo (on-policy) vs. Monte Carlo (off-policy): Monte Carlo methods for updating value estimates utilizing whole episodes worked differently depending on the policy type. The Monte Carlo on-policy agent achieved a 41 per win rate with y=0.8. However, its performance dipped as discount factors increased. While Monte Carlo on-policy can perform well over shorter time horizons, its reliance on episodic returns may make it difficult to anticipate for the long run.

The Monte Carlo off-policy agent demonstrated greater adaptability, achieving a peak victory rate of 40 per at y=0.8 and an average win rate of 19.37. Because of its off-policy nature, this agent may use significance sampling to better assess the value of the target policy, resulting in more robust learning outcomes even when taught with diverse 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.

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%	19.29	36%	9.19
0.8	36%	8.41	39%	12.65
0.9	40%	16.94	39%	17.28
0.99	38%	13.69	39%	27.02

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

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	38%	11.03	40%	28.06
0.8	38%	7.92	36%	12.25
0.9	38%	17.15	41%	21.25
0.99	42%	25.17	41%	24.89

TABLE IV
TABLE 4: MONTE CARLO (ON-POLICY) AND MONTE CARLO (OFF-POLICY) COMPLETE CARD COUNT PERFORMANCE

3) Analysis:

a) SARSA (on-policy) vs. TD(0) (off-policy): table1I The SARSA agent achieved win rates ranging from 36 to 40 per , with the highest average win rate of 19.29 at y=0.7. However, in the Complete Card Count scenario, SARSA struggles to maximize wins.

In contrast, The TD(0) agent beat SARSA, with win rates up to 39 per at y=0.8 and y=0.9, and an average win rate of 27.02 per at y=0.99.This indicates TD(0)'s ability to leverage off-policy updates for successful strategy learning, corroborating the widely acknowledged benefits of off-policy techniques in dealing with different exploration tactics and robust policy adjustments.

b) Monte Carlo (on-policy) vs. Monte Carlo (off-policy): In the Complete Card Counting System, the Monte Carlo on-policy agent consistently obtained win rates of 38 per at y=0.7 and y=0.8, with a significant average win rate of 25.17 at y=0.99. These results show that Monte Carlo on-policy may successfully employ episodic returns to optimize strategies in a card counting context.

The Monte Carlo off-policy agent has high win rates, peaking at 41% at y=0.9 and y=0.99, with the greatest average win rate of 28.06% at y=0.7. This shows that off-policy updates allow this agent to efficiently modify its strategy depending on the total card count, improving 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 :

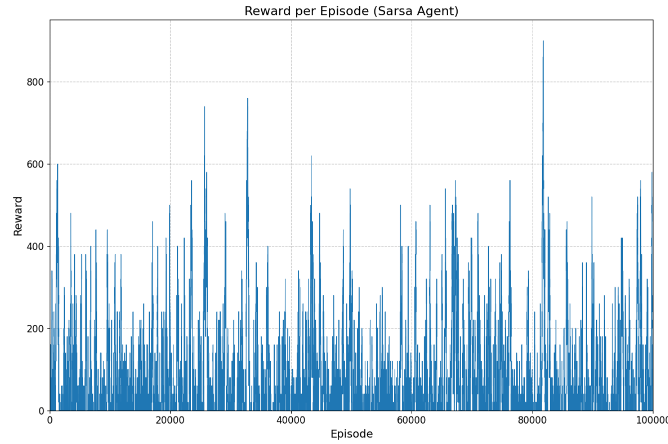

Fig. 1. SARSA (on-policy) Basic Strategy

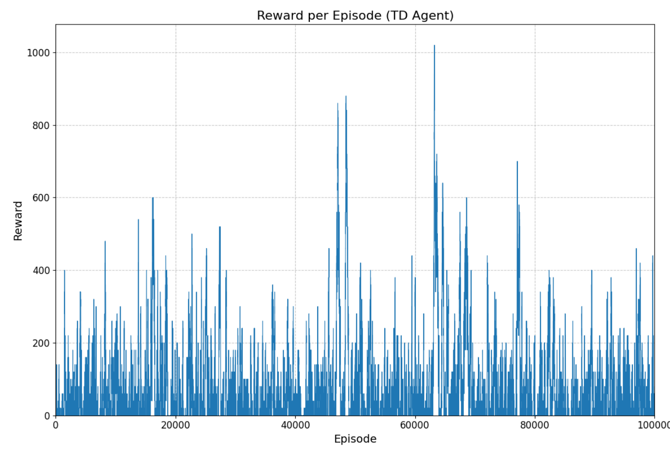

Fig. 2. TD(0) off-policy Basic Strategy

a) SARSA (On-Policy) Reward per Episode Plot: Based on the Fig.1. The Sarsa agent's rewards fluctuate significantly all over 100,000 episodes, ranging from 0 to 900. The pattern is defined by numerous low rewards and infrequent high peaks that do not improve over time. This irregular behavior shows that the agent is unable to settle its policy and may be extremely sensitive to the exploration-exploitation balance.The Sarsa agent occasionally receives big rewards, the agent does not regularly profit on these encounters. The algorithm's performance in this context is called into doubt because there is no evident learning advancement throughout environment.

b) Temporal-Difference (TD) Agent: Fig.2 shows The TD agent's rewards normally vary from 0 to 600, with unusual peaks approaching 1000. Later episodes show a slight tendency of increasingly frequent medium-to-high rewards, indicating incremental development. Even with oscillations, the distribution looks to be more balanced and consistent than the Sarsa agent's. Exceptional peaks reflect the TD agent's ability to execute to the highest level. This minor increase in consistency reflects the TD algorithm's greater capacity to learn from events and fine-tune its policy over time .

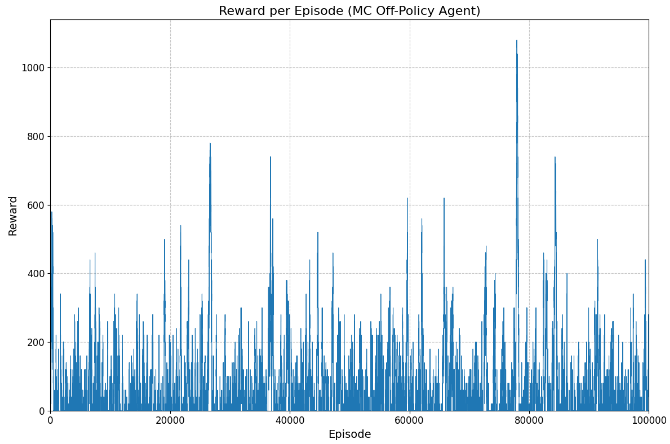

Fig. 3. Monte Carlo off-policy Basic Strategy

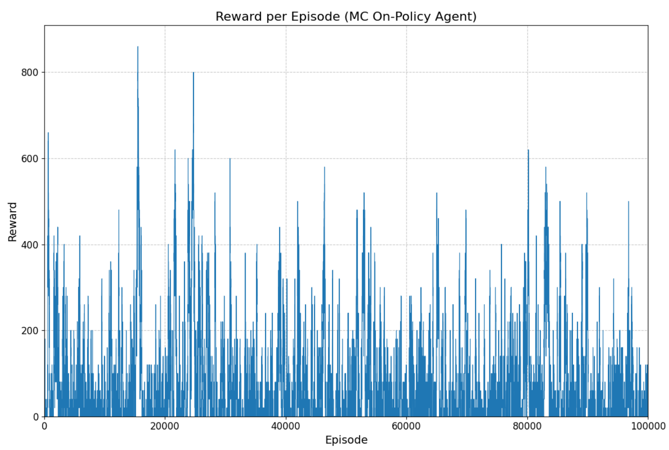

Fig. 4. Monte Carlo on-policy Basic Strategy

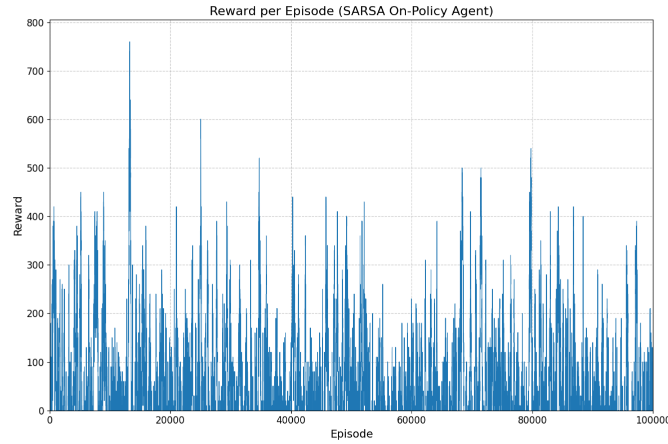

Fig. 5. SARSA(on-policy) Complete Card Count

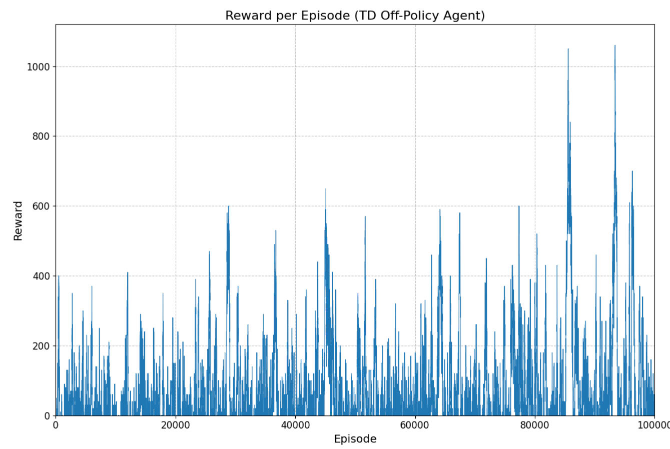

Fig. 6. TD agent Complete Card Count

.

c) Mone-Carlo off-policy: Fig. 3 shows that The MC Off-Policy agent's rewards fluctuate significantly, ranging from 0 to over 1000 every episode. This plot shows frequent low to mid-range rewards with occasional high peaks, implying that the agent discovers high-reward techniques but does not consistently use them. Over 100,000 instances, there are several spikes but no discernible rising trend, indicating that the agent is struggling to develop a stable policy.

d) Monte Carlo on-policy: : Fig. 4. shows that MC On-Policy agent has a more constant reward pattern, ranging from 0 to over 800. The distribution shows fewer severe peaks and more reasonable rewards, implying better exploration-exploitation balance and consistent performance. High-reward experiences are becoming less common, indicating a progressive recovery. The steady reward distribution and increased frequency of moderate awards imply that the MC On-Policy agent learns more effectively from its experiences.

2) Complete Card Count System:

a) SARSA (on-policy): In the Complete Card Counting context, Throughout the 100,000 episodes, the SARSA On-Policy Agent's reward pattern has remained stable, ranging from 0 to roughly 700. Rewards vary within this range, with frequent peaks around 500-700. While there are no particularly large rewards, The Fig. 5 shows rather constant and consistent performance, implying that the agent is successfully learning from its experiences and refining its policy over time.The consistent reward range and frequent increases demonstrate the agent's capacity to use learnt tactics while also exploring and refining its policy.

b) Temporal-Differenc TD Agent (Off-Policy): In comparison, the TD Off-Policy Agent's reward pattern is very variable, with payments ranging from 0 to roughly 1000 each episode. The Fig. 6 reveals many low payouts, broken by big peaks of up to 1000. However, there is no apparent rising trend, and the high awards seem to be episodic rather than indicative of long-term development.The lack of a continuous reward pattern shows the agent is unable to effectively utilize its knowledge to converge on an optimal strategy, resulting in unstable and unpredictable performance over time.

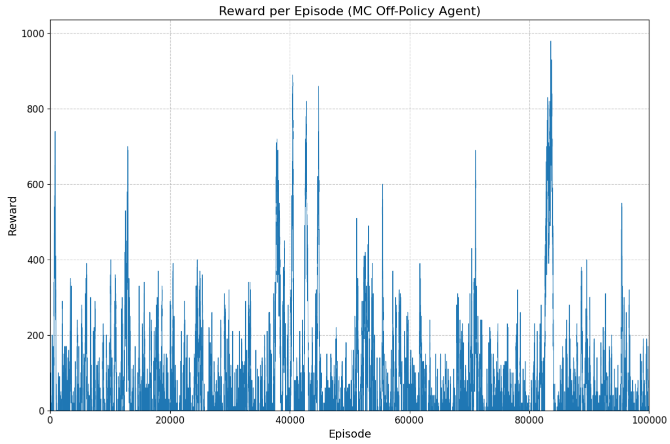

Fig. 7. Monte Carlo off Policy Complete Card Count

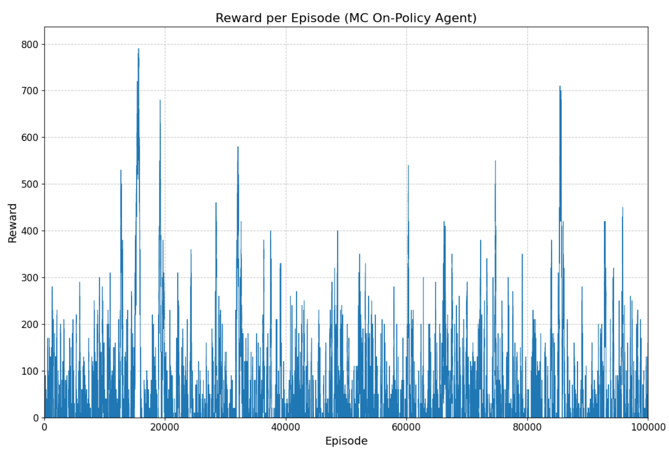

Fig. 8. Monte Carlo on Policy Complete Card Count

c) Monte Carlo Off-Policy: The Fig. 7. shows that in the Complete card counting environment, The reward pattern for the MC Off-Policy Agent is highly variable, with incentives ranging from 0 to about 1000 each episode. The figure indicates numerous low to mid-range rewards, punctuated with occasional high peaks of up to 1000. However, there is no discernible rising trend or continuous improvement across the 100,000 instances. The agent's significant variability and lack of a consistent reward pattern indicate that it has difficulty learning an effective and stable policy from its experiences. While it sometimes stumbles across high-reward tactics, it fails to consistently exploit and build upon those techniques, resulting in irregular and unpredictable performance over time.

d) Monte Carlo On-Policy: The Fig. 8. shows that in the Complete card counting environment, the reward pattern for the MC On-Policy Agent appears to be more regular and steady, with incentives ranging from 0 to about 800. In comparison to the Off-Policy Agent, the reward distribution exhibits a larger frequency of intermediate payouts and fewer severe peaks. This implies that the On-Policy Agent strikes a better balance of exploration and exploitation, resulting in more consistent performance. While there are still occasional big rewards, they are becoming less frequent, showing that the agent's policy is gradually being refined based on its experiences. The more consistent reward distribution and higher frequency of moderate incentives indicate that the MC On-Policy Agent is more adept at learning from mistakes and refining policy over time..

IV. Conclusion

In this study, I conducted an in-depth comparative analysis of four reinforcement learning algorithms—SARSA (On-Policy), Temporal Difference (TD) off Policy, 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. In the Basic Strategy environment, the TD(0) agent regularly beats the SARSA agent. The TD(0) agent's higher peak and average win rates demonstrate the benefits of off-policy methods in effectively learning optimal tactics via varied exploration methodologies. Similarly, Monte Carlo off-policy agents show higher flexibility and robust learning results than their on-policy counterparts, which frequently struggle with long-term strategy stability. Both the SARSA and TD(0) agents perform better in the Complete Card Count environment than in the Basic Strategy environment. However, TD(0) agents continue to outperform SARSA agents by utilizing off-policy updates to attain greater win rates and more consistent strategy refining. Monte Carlo off-policy agents also perform better, with greater peak and average win rates, demonstrating their capacity to adapt and improve tactics over time.

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 plot analysis backs up these conclusions. SARSA agents exhibit variable reward patterns with occasional peaks, implying policy stability challenges. In contrast, TD(0) agents have more balanced reward distributions and incremental gains, indicating an improved capacity to learn from experiences and fine-tune rules. Monte Carlo off-policy agents exhibit high reward fluctuation, implying difficulties in sustaining consistent tactics; Monte Carlo on-policy agents exhibit more stable reward patterns and incremental policy refinement.

Overall, my research demonstrates the better efficiency of off-policy approaches, such as TD(0) and Monte Carlo off-policy, in learning optimum strategies and increasing performance over time. These techniques benefit from comprehensive review and strong policy modifications, which leads to higher win rates and more consistent decision-making procedures. While on-policy options can be somewhat profitable, but often encounter challenges with long-term plan stability and effective predictions. 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.