

From Exploration to Exploitation: Applying Reinforcement Learning to Blackjack

Prahas Hegde

*Master Artificial Intelligence, Department of Computer Science
Technical University of Applied Sciences Würzburg-Schweinfurt
Email: prahas.hegde@study.thws.de*

Abstract—When we talk about blackjack, we immediately think about casinos, money, and cards. The game is renowned worldwide, primarily due to its straightforward rules and potential for high returns. The game is played with one or multiple decks of 52 cards between numerous players and the dealer. The goal is to get a hand less than or equal to 21 or greater than the dealer's hand. The game is lost when the hand value crosses 21 or is less than the dealer's. In this study, we are using reinforcement learning agents to learn strategies namely, basic strategy and complete point count strategy, to maximize player profits over time. This paper uses Q-Learning and SARSA as reinforcement learning agents in the blackjack environment. These agents are trained for several episodes and learn the optimal strategy from their experience. We then compare these strategies in terms of win percentages, rewards per episode etc. In addition to the standard blackjack environment, we're also adding 2 rule variations and comparing the performance for both strategies. This paper discusses how the use of reinforcement learning affects the winning chances of players and compares the results with proper evaluation and visualization.

Index Terms—Artificial Intelligence, Reinforcement Learning, Blackjack, Basic Strategy, Complete point count strategy, Q-Learning, SARSA, Late Surrender, Spanish 21

I. INTRODUCTION

Reinforcement Learning (RL) is a field of machine learning that studies how agents might maximize cumulative rewards by learning to make decisions through trial and error. By interacting with an environment and getting feedback based on their behaviors, RL enables machines to learn. This feedback can take the shape of rewards or penalties. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics, trial and error search and delayed reward are the two most important distinguishing features of reinforcement learning [1]. What is the purpose of reinforcement learning, then? Complex problems that conventional approaches cannot handle may be resolved using RL. The model continuously learns from its environment and can correct errors committed during training. As RL performs well in situations where results are uncertain or change over time, it is highly useful for real-world applications.

A. Introduction to Blackjack

Blackjack is one of the most famous card games played around the world. The game is played between the players and the dealer. The basic blackjack rules remain the same all over but each casino can have their own set of rules to make the game more interesting. All the numeric cards have values equal to their values. King, queen and jack has value 10. Ace on the other hand can have value 11 (soft hand) or 1 (hard hand). The game begins with player's placing their bet. The dealer distributes 2 cards each to all players, called players hand and keep 2 cards for himself with 1 card face up, called dealers hand. The players after receiving the cards can either 'stand' (no more cards) or 'Hit' (ask for 1 more card) depending on their hand. For the player's to win, the hand value should be equal to 21 or be more than dealer's hand. In such case the players receive their bet multiplied by a factor. If the player hand goes above 21 that's a direct loss for the player and he loses his bet. If the player hand is less than the dealer hand, again it's a loss for the player. In case the player hand is equal to 21, then it's called a blackjack and the player receives a higher percentage of his bet. The player directly wins the round in this case unless the dealer too gets a blackjack. In this case the round is a draw. In addition the player's also have options of 'Split' (when player has 2 identical cards, they can play 2 separate hands) and 'Double down' (player ask for additional card doubling the bet value). The interplay of luck and strategy in blackjack allows us to apply reinforcement learning.

B. Getting to know the Strategies

In this paper we are focusing on 2 strategies: The basic strategy and the complete point count system. The Basic strategy are a set of rules defined based on the players hand value and the visible face up card of the dealer. Based on the face up card of the dealer the player can either stand, hit, split or double depending on the player hand value. This strategy doesn't guarantee victory but provides the most efficient way to play the game to increase win percentage.

Card counting systems assign a positive, negative, or zero point value to each card value available. Once a card is dealt, the running count, which starts from 0, is adjusted by that card's point value. Low cards (2-6) usually have positive point values and raise the value of the count, signaling the

increased percentage of high cards in the remaining decks. Conversely, high cards(J, Q, K, A) have negative values and they decrease the count for the opposite reason. System that assign 0 point values to cards (7-9) consider them neutral and they do not affect the running count [2].

The fig. 2 shows the basic strategy for various cases [4].

The chart is a comprehensive guide for Blackjack strategy, organized into several sections:

- DEALER UPCARD** (Top row): Shows dealer upcard values from 2 to A.
- HARD TOTALS** (Left column): Player total values from 2 to 17.
- PAIR SPLITTING** (Column header): Indicated by a yellow background for certain pairs.
- SURRENDER** (Column header): Indicated by a green background for totals 8-10.
- KEY** (Bottom right): A legend defining symbols used in the chart.
- DEALER UPCARD** (Second row): Shows dealer upcard values from 2 to A.
- SOFT TOTALS** (Left column): Player total values from 2 to 17.
- INSURANCE OR EVEN MONEY: DON'T TAKE** (Section title): A table showing whether insurance or even money bets should be taken based on dealer upcard and player soft total.
- DEALER UPCARD** (Third row): Shows dealer upcard values from 2 to A.
- KEY** (Bottom right): A legend defining symbols used in the chart.
- DEALER UPCARD** (Fourth row): Shows dealer upcard values from 2 to A.
- KEY** (Bottom right): A legend defining symbols used in the chart.

Fig. 1. The basic strategy including all of the situations

C. Reinforcement Learning Agents

In this paper we have used 2 reinforcement learning algorithms, namely Q-Learning and SARSA.

Q-Learning is an off-policy, model-free reinforcement learning method that facilitates interaction and allows the agent to learn from its surroundings in order to make the best choices. With Q-values indicating the optimal course of action in the specified state, the agent creates a Q-table. The Q-table is subsequently updated by the agent in accordance to the feedback.

SARSA is a model-free, on-policy which helps the agent interact and learn from the environment and make optimal decisions. Unlike Q-learning it doesn't depend on the best possible actions, rather it allows the agent to explore and tends to converge to a conservative optimal policy.

A. Blackjack Environment for Basic Strategy

The blackjack environment consists of blackjack class that governs the game. The parameters are used to specify number of 52 card decks and the winner payout. The createDeck methods generates the specified number of well shuffled decks and ensure random card distribution. The reset method is used to generate new hand and also deals with immediate blackjack providing instant reward. The getState method represents the state of the environment needed by the RL agent. It returns a tuple of: player current hand value, dealer up card value and flags for usable ace(differentiate hard/soft hands), double down(can the player use double down) and split(can the player split). This method is used by the agent for its learning. The step method consists of the action taken by the agent which includes: 0:stand(no more cards), 1:Hit(take additional card), 2:double(double the bet and take additional card) and 3:split(if player has 2 similar cards than play them as 2 separate hands). The player gets +1 if he wins(player hand more than dealer hand), -1 if he loses(player busts or dealer hand more than player), 0 for a push(player and dealer have equal hands), -2 if player loses after doubling down and +/-2 rewards if hands are split. The dealer on the other hand follows the fixed casino rules. The dealer must hit until his hand value is greater than equal to 17. The game doesn't allow soft 17 condition for the dealer.

In Basic strategy, players stand on high hard totals (17 or more) and hit on low hard totals (8 or less). The dealer's upcard plays a major role in determining whether to hit or stand for intermediate hard totals (12–16), with standing being recommended against weak dealer upcards (2–6) and hitting against powerful ones (7–Ace).

Players are typically more aggressive with soft hands because of the Ace's flexibility; they frequently hit or double down on lesser soft totals because there is no chance of busting on the following card. Almost invariably, soft 19 and 20 are stood on. Certain pairs, like Aces and 8, are always split, whereas other pairs, like 10 and 5, are never split. Only specific dealer upcards can split other pairs [3].

B. Blackjack Environment for complete point count system

The blackjack class is enhanced for implementing the card counting system. runningCount tracks value of the cards and is initialized to 0. cardsDealtCount tracks how many cards have been played. Method getCardCountValue is responsible for the point count. It assigns -1 to high cards(J, Q, K, A), +1 to low cards (2-6) and 0 to cards 7-9. The getState method is updated to include parameter binnedTrueCount, containing card counting information. The binnedTrueCount is obtained by dividing runningCount by number of remainingDecks. The avtion space and reward remain the same as that of the basic strategy. Similarly the dealer play remains according to the predefined rules.

C. Q-Learning

The QLearningAgent class is used to implement Q-Learning algorithm. It has several parameters, like actions tuple(stand,

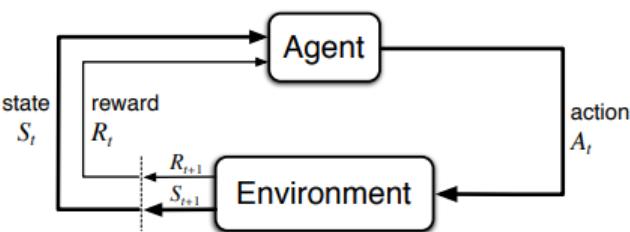


Fig. 2. Reinforcement Learning block diagram

II. METHODOLOGY

This section explains how the reinforcement learning agents are trained using the basic and point count strategies. It describes RL agents, the blackjack environment, and rule modifications and how they affect agent performance.

hit, double, split), learning rate α (The extent by which Q-values are updated), Discount factor γ (present value of future reward), epsilon ϵ (probability of exploration), q table(with q values), epsilon decay rate, action map. The chooseAction method balances exploration and exploitation by implementing ϵ greedy policy. The learn method is where the update rule is applied and the Q-Value is updated based on eq. (1).

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right] \quad (1)$$

The decayEpsilon method is used to periodically reduce the value of ϵ . This ensures the agent gets enough experience and q table becomes accurate. getPolicy method allows us to extract the learned policy. It defines action with highest q value for each state in q table. The Q-learning algorithm is an excellent method for approximating an optimal blackjack strategy because it allows learning to take place during play. This makes it a good choice for the blackjack problem domain. Blackjack is easily formulated as an episodic task, where the terminal state of an episode corresponds to the end of a hand.

D. SARSA

The SARSAAgent class implements the algorithm. It initializes the parameters like: learning rate α , discount factor γ and epsilon ϵ for controlling exploration-exploitation trade off. It uses and ϵ greedy policy to either explore or exploit. The Q-Value is updated based on the eq. (2).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2)$$

On-policy algorithms, such as SARSA, strike a balance between exploration and exploitation strategies by choosing random actions with a chance of selecting non-greedy actions. The decayEpsilon method is used to periodically reduce the value of ϵ and shifts the agent's focus to exploitation.

E. Spanish 21

One of the variations in blackjack is Spanish 21. We use a deck containing 48 cards as all the 10 are removed. All the numeric cards have same value as their face value similarly J, Q, K have value 10 with A being either 1 or 11. The goal of the player is to get hand value greater than the dealer. If the player hand value is equal to 21 it's a blackjack and in Spanish 21, the player wins the hand even if the dealer has 21. So it becomes advantageous for the players. The dealer is allowed to stand on soft 17, while other rules are similar to standard game. Split is allowed, doubling after split is also allowed and players can draw multiple cards after split. The players also receive side bonuses if the dealer's up card matched the player's card and when there is a rare 7-7-7 between player and dealer, with dealer having 1 out of 3, 7's.

The Spanish21Env class is used to define the rules of the game. It initializes the number of decks(48 cards), game payout and bonus payouts for side challenges. We define the win case for player blackjack even if the dealer hand is 21. The dealer rules are similar to the previous cases but with stand option on soft

17. The getState method takes in parameters such as player hand value, dealer up card, flags for Ace, split and double. This is essential for the agent to learn. In the paper we are implementing the Spanish 21 with both basic and point count strategy using Q-Learning as RL agent.

F. Late Surrender

Late surrender is a variation in blackjack, which allows a player to end his hand and lose half the bet but the dealer should check for a natural blackjack. If the dealer has a blackjack, then the surrender is not valid and player loses all his bet. For adding this variation we modify the BlackjackEnv class. We pass in an additional parameter, allowLateSurrender, which is a boolean. The reset method includes a canSurrender flag to ensure surrender takes place at the start after card are dealt with. The agent in addition to stand, hit, double, split has an option to surrender. When the agent selects late surrender, the reward received will -0.5 when conditions for the surrender are True. The getState method will get an additional parameter canSurrender indication if surrender is possible or not.

III. EXPERIMENTATION AND EVALUATION

In this section we will be going through the results and observations from playing Blackjack using different strategies and reinforcement learning agents. Compare them using evaluations and visualizations.

A. Basic Strategy with Q-Learning

The agent is made to learn the basic strategy for blackjack using Q-Learning. The agent is trained for 1000000 episodes and evaluated on 100000 episodes. After training, we get the expected profit per hand = -0.0123, win rate = 43.50%, loss rate = 47.87%, draw rate = 8.63% also player natural blackjacks = 4462, and dealer natural blackjacks = 4715. The training was done with a learning rate of 0.01. The results show agent's successful learning of basic strategy.fig. 3 depicts the average reward during each episode.

TABLE I
BASIC STRATEGY WITH Q-LEARNING SUMMARY

Strategy	Episodes	Win	Loss	Draw
Basic	1000000	43.50%	47.87%	8.63%

B. Complete point count system with Q-Learning

The agent is made to learn the point count strategy for blackjack using Q-Learning. The agent is trained for 2000000 episodes and evaluated on 200000 episodes. After training, we get the expected profit per hand = -0.0164, win rate = 43.06%, loss rate = 48.22%, draw rate = 8.72% also player natural blackjacks = 9036, and dealer natural blackjacks = 9181. The training was done with a learning rate of 0.01. The results show agent's successful learning of point count strategy.fig. 4 depicts the average reward during each episode.

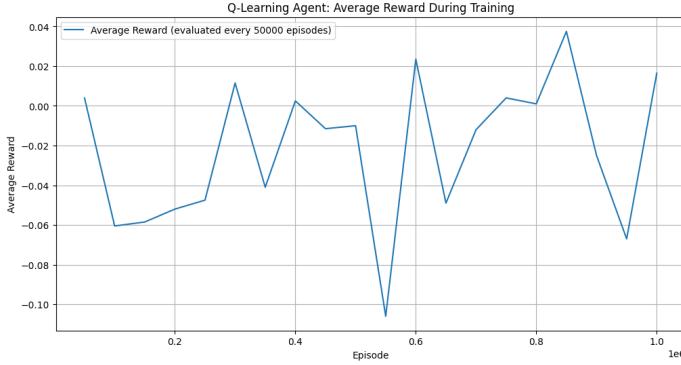


Fig. 3. Q-Learning average reward during training (Basic Strategy)

TABLE II
COMPLETE POINT COUNT SYSTEM WITH Q-LEARNING SUMMARY

Strategy	Episodes	Win	Loss	Draw
point count	2000000	43.06%	48.22%	8.72%

C. Complete point count system with SARSA

The agent is made to learn the point count strategy for blackjack using SARSA in order to check if there is any improvement in its performance from Q-Learning. The agent is trained for 2000000 episodes and evaluated on 200000 episodes. After training, we get the expected profit per hand = 0.0586, win rate = 45.34%, loss rate = 45.88%, draw rate = 8.78% also player natural blackjacks = 8900, and dealer natural blackjacks = 8195. The training was done with a learning rate of 0.01. The results show agent's successful learning of point count strategy.fig. 5 depicts the average reward during each episode.

As we can see from table I,table II,table III, the win rate of Basic strategy with Q-Learning(43.50%) is slightly more than the point count strategy with Q-Learning(43.06%). While basic strategy is meant to reduce losses, the point counting strategy aims to create a positive outcome. The difference in winning rates shows that, with the current training setup and

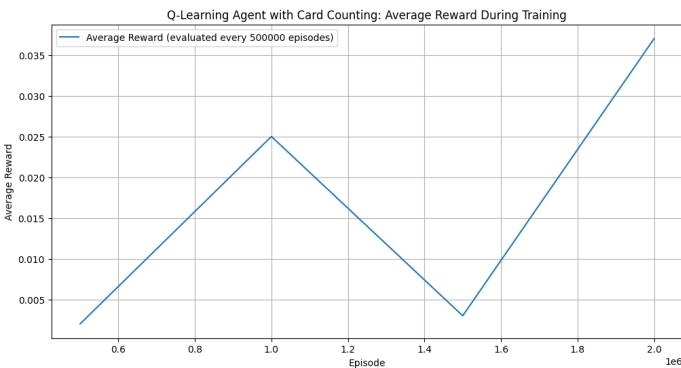


Fig. 4. Q-Learning average reward during training (card counting)

TABLE III
COMPLETE POINT COUNT SYSTEM WITH SARSA

Strategy	Episodes	Win	Loss	Draw
point count	2000000	45.34%	45.88%	8.78%

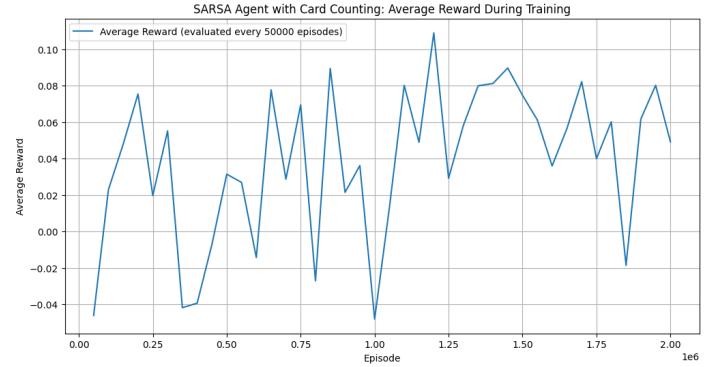


Fig. 5. SARSA average reward during training (card counting)

environment, the Q-learning agent might have better reached the best version of basic strategy than the more detailed and complex point-counting strategy. More research, perhaps with additional training rounds or improved adjustments to the learning settings for the card counting agent, could help determine whether the advantages of point counting can be fully leveraged in this reinforcement learning system. The point count strategy with SARSA outperforms both with win rate of 45.34%.

D. Spanish 21 with Basic and point count strategy

The Spanish 21 variation for blackjack is implemented in the environment and the Q-Learning agent is used to learn the game using both basic and point count strategy. The agent performance is later compared. It is trained for 1000000 episodes and evaluated on 100000 episodes. The agent performance is computed with and without the variation for both the strategies. This helps us determine whether the variations were really helpful or profitable for the players and how it affects the agent performance.fig. 5 and fig. 6 depicts the average reward during each episode.

TABLE IV
SPANISH 21 FOR BASIC AND POINT COUNT SYSTEM

Strategy	Avg Reward	Win	Loss	Draw
basic with Spanish 21	-0.0332	38.21%	53.24%	8.54%
basic	-0.0418	37.19%	54.98%	7.83%
point count with Spanish 21	-0.0406	38.43%	53.23%	8.34%
point count	-0.0600	36.79%	55.41%	7.81%

As we can see from table IV, fig. 6 and fig. 7, the Spanish 21 variation improves the agent performance in both the strategies. We can see that the the variation has a higher win

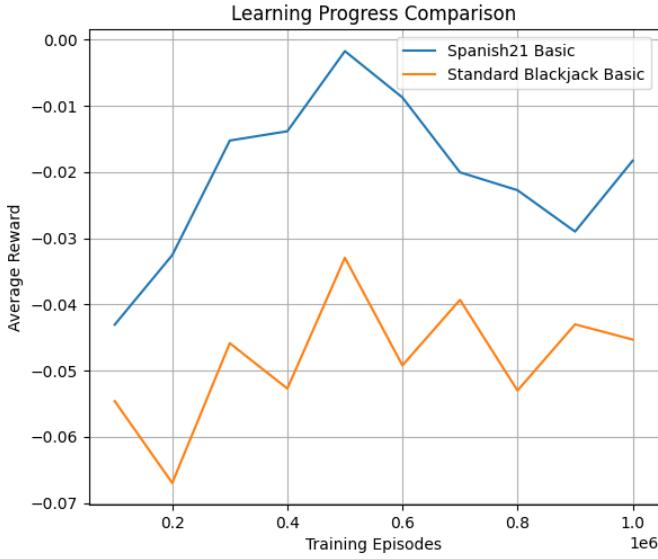


Fig. 6. Average reward during training(with/without Spanish 21) using Basic Strategy



Fig. 7. Average reward during training(with/without Spanish 21) using point count Strategy

rate and also higher reward than the standard one's. The Spanish 21 with it's additional bonuses and player blackjack advantage has a better performance. We can also see that point count system with variation performs the best out others.

E. Late Surrender with Basic and point count strategy

The late surrender option in blackjack is added of the game environment, and a Q-Learning agent is used to learn how to play the game using both basic strategy and a point count strategy. The agent's performance is then compared. It is trained for 1,000,000 rounds and tested on 100,000 rounds. The agent's results are measured with and without the late surrender rule for both strategies. This helps us find out if

the rule actually helps or benefits players and how it impacts the agent's performance. fig. 8 and fig. 9 depicts the average reward during each episode.

TABLE V
LATE SURRENDER FOR BASIC AND POINT COUNT SYSTEM

Strategy	Avg Reward	Win	Loss	Draw
basic with Surrender	-0.0388	37.23%	55.22%	7.56%
basic	-0.0373	42.86%	47.78%	9.35%
point count with Surrender	-0.0250	38.90%	53.57%	7.54%
point count	-0.0198	43.07%	48.31%	8.62%

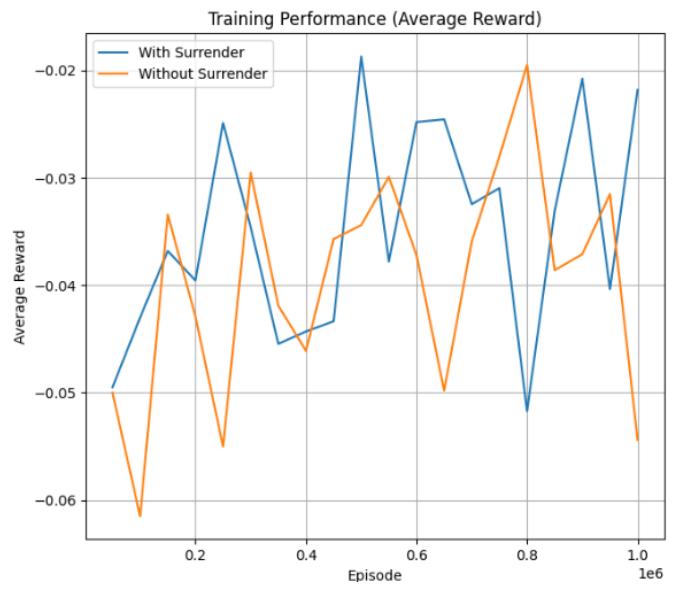


Fig. 8. Average reward during training(with/without Late Surrender) using Basic Strategy



Fig. 9. Average reward during training(with/without Late Surrender) using point count Strategy

As we can see form table V and fig. 8 and fig. 9, the late surrender variation reduces the win rate of the agent. The agent performs far better using the standard strategies without the variation. The surrender option allows the agent to minimize

the losses in unfavorable conditions and you can see that even if there is large difference in win rate the average rewards vary by little amount. The point count system with surrender performs slightly better than the basic strategy.

As seen in fig. 10, surrender is used when the true count is around 0 or in the middle, potentially reducing losses. At the extreme ranges of true count surrender is not used as it is not optimal, the agent will either stand, hit, split or double.

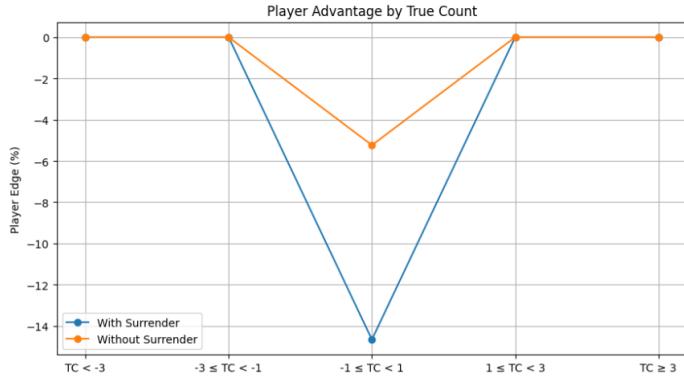


Fig. 10. Player advantage by True Count (with/without Late Surrender) using point count Strategy

IV. CONCLUSION

In this paper we are applying the reinforcement learning to the game of blackjack. We are using 2 RL agents namely, Q-Learning and SARSA. We are training the agents with 2 strategies: The basic strategy and complete point count system. When we train the Q-Learning agent using basic strategy it gave a win rate of 43.50%. The same agent with the complete point count system gave a win rate of 43.06%. The point counting technique seeks to produce a favorable result, whereas the basic strategy is intended to minimize losses. The disparity in win rates indicates that the Q-learning agent may have more successfully attained the optimal version of the basic strategy than the more intricate and sophisticated point-counting method within the current training configuration and environment. When we implemented the point count system using SARSA we got a win rate of 45.34%, making it the optimal among the 3 implementations.

We then introduced 2 variations in the blackjack environment: Spanish 21 and Late Surrender. In the case of Spanish 21, the agent was trained on basic and point count strategy using Q-Learning. Point count system with Spanish 21 showed the best performance (38.43%), followed by basic strategy with Spanish 21 (38.21%). The implementations without the variations were slightly less. We can say that Spanish 21 with its extra bonuses and player blackjack advantage gives higher win rate and thus higher profit for the player. The late surrender variation gave less win rate with both strategies, the highest given by point count system (38.90%). The strategies without the variation showed tremendous improvement getting win rate of 42.86% (basic) and 43.07% (point count). Even with huge difference in win rate the average reward varied slightly for

both strategies, showing that surrender helps to minimize the loss when the outcome is unfavorable.

These implementations indicate that, despite a win rate ranging from 37% to 45%, there is still significant room for improvement. To improve performance, we can employ sophisticated techniques like deep Q networks and Bayesian Q-learning algorithms, which delve deeper into the area. There is a lot of room for future research in this area of reinforcement learning applied to blackjack.

ACKNOWLEDGMENT

Firstly, I would like to thank Prof. Dr. Frank Deinzer. It was a pleasure to attend his course on "Reasoning and Decision Making under Uncertainty". The course gave me a fundamental understanding of Reinforcement learning. I would like to thank Edward O. Thorp and his book "Beat the Dealer" for explaining me all about blackjack and strategies involved in it. I'm grateful to my parents for allowing me to pursue my MSc in AI and always supporting me. I'm also grateful to my university THWS for all the support and providing all crucial resources for the completion of my paper. I would also like to mention the tools used in the making of this paper namely: overleaf, quill bot, google collab, grammerly and github. All have contributed significantly towards the completion of the research paper.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
- [2] M'ozes Vid'ami1, L'aszl'o Szil'agyi1,2,3, and David Iclanzan1, Real Valued Card Counting Strategies for the Game of Blackjack.
- [3] E. O. Thorp, *Beat the Dealer: A Winning Strategy for the Game of Twenty-One*. New York, NY, USA: Vintage, 1966.
- [4] Xinyi Cai, *The House Edge and the Basic Strategy of Blackjack*
- [5] Charles de Granville, University of Oklahoma, Applying Reinforcement Learning to Blackjack Using Q-Learning