university of groningen / faculty of science and engineering / artificial intelligence

Skull and Roses: Multi-agent Reinforcement Learning approach of modelling Bluffing and Theory of Mind

Bachelor's Project Thesis

Alexandru-Dumitru Ditu, s4004027, a.ditu@student.rug.nl,
Supervisors: Dr. Harmen de Weerd

Abstract: This study explores the possible benefits of using a bluffing strategy in the bidding phase of the game Skull and Roses, an imperfect information game that has many similarities with other games such as Poker. By simulating various game scenarios using Python and reinforcement learning agents, the research aims to determine whether bluffing provides a statistical advantage to the player when compared to strategies that do not employ bluffing. Essential concepts such as game theory, reinforcement learning, Q-Learning and Theory of Mind (ToM) are discussed in order to provide a theoretical foundation for the study. I hypothesize that bluffing will outperform the non-bluffing strategies or at the very least provide similar performance to them. The results were extracted over a total of 100 individual simulations, containing 1000 game rounds each, for each game scenario. After analysis, the results show that the agents that are using the bluffing strategy are slightly outperforming those that do not, demonstrating a statistically significant advantage. Therefore, this outcome is in line with the hypothesis. However, a key discovery has been made in the testing phase: bluffing is only beneficial when used alongside specific card layouts such as the last placed card being a Skull. This hints that there is a very close relation between bluffing and the first phase of the game. These findings contribute to a deeper understanding of strategic decision-making in complex, imperfect information games. Further research is needed in order to find out if the bluffing strategy can be further improved by implementing Reinforcement Learning in all game phases.

1 Introduction

This study explores the strategic decision-making process in imperfect information games, with a particular focus on the concept of bluffing. The study uses the game Skull and Roses (Marly, 2011) as a case study to investigate whether bluffing can provide a statistical advantage to players. By simulating various game scenarios with AI agents employing different strategies (regular Q-Learning, enforced bluffing and forbidden bluffing), this research aims to contribute to a deeper understanding of the effectiveness of bluffing in strategic games. The following sections will introduce essential concepts such as game theory, reinforcement learning, and Theory of Mind (ToM), which form the foundation for analysing and developing strategies in Skull and Roses. Lastly, the results of the experiment will be showcased and discussed in order to reach a conclusion.

1.1 Introducing Concepts

1.1.1 Game Theory

The mid 20th century saw the birth of a new branch of applied mathematics, called game theory. Its purpose was to analyse decision-making processes found in games such as Chess in order to model strategies that could be applied in the real world in various fields. As a result of the emergence and development of this novel theoretical framework many fields saw considerable advancements (Dufwenberg, 2011). Some examples of fields that benefited from advancements made in game theory are computer science, economics and biology.

The beginning of experimentation with Artificial Intelligence in games happened roughly at the same time as the field of game theory was gaining traction, the mid 20th century, when scientists attempted to simulate a chess player. A few notable instances of AI being successfully applied in games are IBM's DeepBlue model that managed to win against the chess world champion in 1997 (Campbell et al., 2002) as well as Google's AlphaGo model that successfully won against an experienced Go player in 2016. Recently, a new version called AlphaGo Zero defeated the previous AlphaGo version without domain knowledge or human guidance. (Silver et al., 2017)

1.1.2 Perfect & Imperfect Information

Games can be broadly categorised into two types: perfect information games and imperfect information games. The first one refers to games where all players know all the information about the current state of the game; such games include Chess and Go, where the players can view the whole board at any moment, including the opponent's pieces. The only unknown in perfect information games is the strategy that each player employs. In contrast, imperfect information games refers to games where not all information is disclosed to all players (such as the types of cards each player is holding, their decisions and strategy etc). Among others, such games include Poker and Skull and Roses; in both of these games players will not know each other's cards or decisions until the end of the round when the cards are revealed. Implementing an AI model on this type of game poses significant challenges compared to a perfect information game (Schofield & Thielscher, 2019). This is happening because the complexity of an imperfect information game is much higher. However, this complexity is the very reason why their study is important. This is because these types of games more closely resemble the complexity found in real life scenarios from various human-driven fields and industries.

1.1.3 Reinforcement Learning

Reinforcement learning (RL) is a branch of machine learning that emerged as an effective method of implementing Artificial Intelligence in various types of problems, including imperfect information games. When reinforcement learning (RL) is implemented correctly, agents acquire the ability to make decisions by executing actions and receiving rewards upon the completion of the specified task, thereby incrementally enhancing their strategies through a process of trial and error (Kaelbling et al., 1996). If the task was completed successfully (or better than in the last attempt), then the reward is higher. A lower reward or a negative reward is awarded if the task was not completed successfully (or worse than in the last attempt). This iterative process encompasses essential elements including agents, states, actions, rewards, and policies.

Q-Learning is a widely used reinforcement learning algorithm that enables agents to learn the best strategies by updating a Q-Table, which estimates the value of actions in specific situations (also called states) (Jang et al., 2019). Furthermore, the correct balance between exploration (trying new actions) and exploitation (choosing the best known actions) is crucial for efficient training and can be set through the parameter "epsilon". A higher epsilon value will promote exploration, meaning that the agent will attempt to find new (and potentially better) strategies. A lower epsilon value will promote exploitation, meaning that the agent will take advantage of the information it has already gathered about strategies and use the best one found so far. Exploration is especially necessary in multi-agent settings because the values of actions can change over time due to the dynamic interactions between agents (Buşoniu et al., 2008).

1.1.4 Theory of Mind

An essential concept related to strategic decision-making in games is Theory of Mind (ToM). ToM refers to the ability to attribute mental states such as beliefs, intents, and desires to another agent (Yoshida et al., 2008).

In the context of games, ToM enables players to predict and interpret the actions of their opponents by trying to mentally simulate what the opponent is thinking and take advantage of that. This is particularly important in games involving deception, like Skull & Roses, where players must

anticipate how their actions will be perceived by others. For example in Poker, if Theory of Mind is applied correctly in the scope of bluffing, a player could bet everything in hopes of demoralising its opponent and forcing them to fold.

Theory of Mind level zero refers to the scenario when a person is making decisions based on the behavior of another person (as exemplified in Figure 1.1).

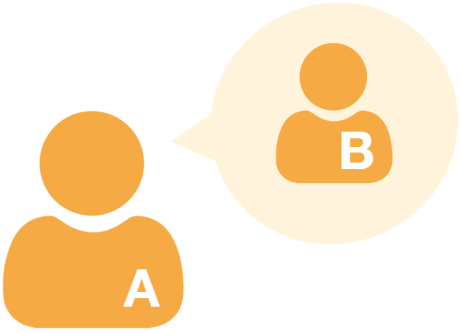

Figure 1.1: Theory of Mind - Level 0

Continuing the ToM hierarchy, starting with ToM 1, recursive reasoning becomes paramount. Theory of Mind level one entails person 'A' thinking about what person 'B' will think of their actions, and making a decision based on that (Figure 1.2). The bluffing we will be discussing in this research falls under the ToM level 1 category. More precisely, in the process of bluffing, player 'A' (the bluffing player) will think about how player 'B' will interpret their actions and how that will reflect in the choices that player 'B' will make.

For an ideal bluffing round the following scenario would play out: player 'A' is placing a bet higher than their maximum safe bid, player 'B' will assume that player 'A' is playing a safe bet, and therefore increase the bid once again, effectively falling in the trap laid by player 'A'.

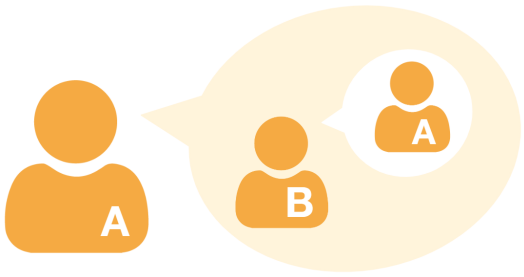

Figure 1.2: Theory of Mind - Level 1

ToM level 2 (Figure 1.3) takes bluffing to an even more advanced level: continuing the Poker example provided earlier, if we know we successfully bluffed in the past by betting everything, when playing with the same opponent in the future, we can bet everything again when the cards are in our favour. This way, the opponent will believe that we are bluffing, but in fact we are not. Both Tom 1 and ToM 2 are forms of recursive reasoning that can be useful when modelling reinforcement learning for the scope of finding the best strategies in a game.

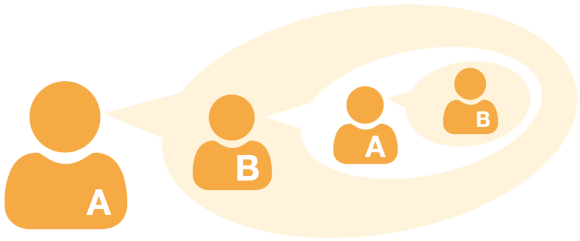

Figure 1.3: Theory of Mind - Level 2

A player that makes use of the bluffing strategy deceives opponents regarding the true worth of their hand or decisions. This approach is thoroughly investigated in card games like poker, where a player's chances of winning can be significantly enhanced with adept bluffing (Guazzini & Vilone, 2013). Bluffing necessitates players to meticulously assess the risk and reward, since they must weigh the possibility of their deception being exposed.

1.2 Skull and Roses

Skull and Roses was designed by Hervé Marly in 2010 and falls under the imperfect information

category of games. It encompasses various mechanics that are similar to the game of Poker, such as bidding and revealing the cards at the end of the round.

As previously stated, it provides a very good foundation for artificial intelligence research due to its complexity. Furthermore, the number of rules is relatively small but at the same time, the game allows space for a high number of possible strategies that could be employed.

Figure 1.4: The cards used in Skull & Roses. Each player starts with a hand of: 1 Skull card and 3 Rose cards.

In Skull and Roses, each player starts with four cards: three roses and one skull (Figure 1.4). The game progresses in three phases: card placement, bidding, and card-flipping. During the card placement phase, players take turns placing one card face down on a personal stack until everyone has placed at least one card. In the bidding phase, players bid on how many cards they believe they can flip without revealing a skull, with the highest bidder becoming the 'challenger'. In the card-flipping phase, the challenger is required to reveal the same number of cards as their bid value, starting by first flipping over all of their own cards and then moving on to the opponent's cards. Revealing a skull results in losing the round, while successfully flipping the required number of cards without revealing any skulls wins the round. This study focuses on the strategies related to the bidding phase, where bluffing plays a crucial role.

Strategies in Skull & Roses don't solely revolve around the bidding phase, however in this study the focus will be on the strategies related to the bidding phase (second stage of the game) because that is where the possibility of bluffing comes into play. While choosing a strategy, players must decide whether to bid conservatively (therefore if they become the challenger, they know for sure that no skull will be revealed from their own stack), risking losing the round if they don't win the bidding war, or to bluff by bidding more than they can safely flip, hoping to influence opponents into taking risky decisions by bidding a higher value. Bluffing in Skull and Roses comes in two variations:

- **Bluffing when the player knows the bid cannot be satisfied**: This happens when a player has placed a skull in their stack but still chooses to bid higher than their maximum safe bid, knowing that if challenged, they will inevitably fail to reveal the required number of safe cards. The goal here is to deceive opponents into believing the player has a higher maximum safe bid than they actually do.
- **Bluffing when the player thinks the bid can be satisfied**: Similar to poker, where a player might bluff with a low-value hand, betting as if they have a high-value hand. In this scenario, the bluffing player does not place down a Skull card but bids a high value hoping to fulfill it.

Overall, bluffing relies on deceiving opponents into believing the player has a stronger position in the current round than they actually do. In a real life scenario, successful bluffing requires an understanding of the opponent's tendencies and the ability to anticipate their responses. However, even a rudimentary version of bluffing could bring to light a difference in performance.

1.3 Scope of the research

Given the importance of bluffing in imperfect information games such as Skull and Roses, this research investigates whether implementing a bluffing strategy provides a statistical advantage to a player compared to a non-bluffing strategy. Specifically, this study aims to determine if players who incorporate bluffing into their

strategy outperform those who do not. Therefore, the research question addressed in this study is: **Does applying a bluffing strategy in a two-player game of Skull and Roses provide a statistical advantage to the player?**

The hypothesis posits that bluffing should result in at least neutral outcomes when compared to non-bluffing strategies and potentially offer a statistical advantage. It is expected that bluffing will either improve or maintain the player's chances of winning when used strategically. To explore this hypothesis, various game scenarios will be simulated using AI agents with different strategies, including bluffing and non-bluffing but also completely forbidding the use of bluffing. The results of these simulations will provide insights into the effectiveness of bluffing in Skull and Roses, contributing to a deeper understanding of strategic decision-making in imperfect information games.

2 Methods

In this section, we describe the methods used to investigate the effectiveness of different strategies in the game of Skull and Roses. The methodology involves creating a detailed simulation of the game, implementing various artificial intelligence (AI) strategies, and conducting experiments to collect and analyse data on the performance of these strategies.

We start by outlining the construction of the game simulation, detailing the classes and parameters involved. This is followed by an explanation of how Reinforcement Learning, particularly Q-learning, is employed to enable AI agents to learn and adapt their strategies over time. Finally, we present the experimental setup, specifying the different scenarios tested and the process used to collect and analyse the results.

2.1 Game Implementation

To simulate the game of Skull and Roses, the Python programming language was used, taking advantage of its object-oriented features. The simulation is built around two main classes: Player and SkullGame. These classes handle the behaviour of the players and the overall game flow.

- Player Class: This class represents each player in the game. It includes methods for placing cards, choosing bids, resetting the environment after each simulation, and updating the Q-learning table based on the outcomes of their actions.

- SkullGame Class: This class manages the sequence of game phases, including card placement, bidding, and card-turning. It also handles game resets and the reward system for Q-learning.

The simulation allows for customization through several parameters, some of which are fixed for the purpose of this experiment:

- FIXED_STARTER (fixed to 0):
  Determines if Player 1 always starts (1) or if the starting player is selected randomly (0).

- ALL_CARDS_BEFORE_BID (fixed to 0):
  Controls whether all cards must be placed before bidding starts (1) or if bidding can start before all cards are placed (0).

- USE_RANDOM_BLUFF_CHANCE (fixed to False):
  If set to True, the bluff chance is randomised; otherwise, a fixed bluff chance is used.

- BLUFF_CHANCE:
  Sets the probability of bluffing (e.g., 15%).

- Use_Softmax_Policy & Use_UCB_Policy (both are fixed to False):
  Enable different exploration strategies like Softmax and Upper Confidence Bound (UCB), respectively.

2.2 AI in Skull and Roses

Reinforcement learning (RL) was used to allow players to learn optimal bidding strategies. I used the Q-learning algorithm because it works well for discrete action spaces like those found in Skull and Roses (Fares Fourati, 2024).

- Q-Learning: Players use a Q-table to store values of state-action pairs. These Q-values are updated based on rewards received after actions and the estimated future rewards.
- Q-Table: A dictionary where keys are state-action pairs and values are the estimated rewards for those pairs.

The reward function is crucial for guiding the learning process. The rewards were designed as follows: for every successful action the reward is +10 and for every failed action the reward is -10. This reward function was achieved after experimenting with different implementations, many of which offered a higher penalty for a failed bluffing attempt. However, this did not lead to different results when compared to a standard reward function, therefore the less complicated version was chosen. The Q-learning implementation can be observed in Listing 1 and is given by the mathematical formula (2.1):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2.1}$$

where:

- $Q(s,a)$ is the Q-value for the state-action pair (s,a)
- α is the learning rate
- r is the reward received after taking action a in state s
- γ is the discount factor
- $\max_{a'} Q(s',a')$ is the maximum Q-value for the next state s' over all possible actions a'

```
1 def update_q_value(self, state, action, reward,
      next_state, max_bid):
2     old_value = self.q_table.get((state, action),
      0)
3     future_rewards = max([self.q_table.get((
      next_state, a), 0) for a in range(1, max_bid +
      1)], default=0)
4     new_value = old_value + self.alpha * (reward +
      self.gamma * future_rewards - old_value)
5     self.q_table[(state, action)] = new_value
```

Listing 1: Python function responsible updating the QTable

The state representation used by the Reinforcement Learning algorithm consists of:

- The number of cards on the player's stack.
- The number of cards on the opponent's stack.
- The number of roses on the player's stack.
- The presence of a skull on the player's stack (boolean).
- The current bid.
- The maximum safe bid.

This particular state representation has been chosen because it aggregates all the important freely available information for each player. This implementation provides the reinforcement learning algorithm with enough information to assess the current state of the game and compare it with previous states. To give more context, the number of cards in the opponent's stack can influence the chance that a Skull has been placed on the table, while the number of roses on the player's stack along with the maximum safe bid and the Skull boolean allows for the system to infer the exact placement of the cards on the player's stack.

Different policies were employed to ensure a comprehensive exploration of the action space:

- Epsilon-Greedy: With a probability epsilon, players select random actions; otherwise, they choose the best-known action. The epsilon value decays over time to shift from exploration to exploitation (Sewak, 2019).
- UCB: This algorithm selects actions based on their upper confidence bounds of estimated rewards, promoting exploration of less-frequented actions (Ye & Chen, 2022).
- Softmax: Actions are chosen probabilistically based on their Q-values, allowing for more nuanced exploration (Syafiie et al., 2004).

Bluffing was integrated by adjusting the bidding process. Players designated as bluffers would consider bluffing based on a specified probability. Furthermore, a secondary condition needed to be met in order for bluffing to go into effect: the

last placed card of the bluffing player had to be a 'Skull'. This decision is meant to more closely resemble a real life scenario (in a real life game of Skull and Roses, a player may choose to bluff only when the cards are placed in a specific order, increasing the chances that the opponent will fall into the 'trap'). The 'Skull' card being on top reflects the best opportunity to bluff because it allows the bluffing to be successful with the lowest opponent bid (the opponent only needs to turn one of the bluffing player's cards in order to lose). This strategy was tested in scenarios where the opponent was relying only on Q-Learning to place bets.

```
function consider_bluff(action, max_bid):
    if not use_random_bluff_chance:
        bluff_chance = predefined_bluff_chance
    else:
        bluff_chance = random value between 0 and 1
    if player_is_bluffing and random value <
     bluff_chance and last card on table is 'skull':
        currently_bluffing = True
        safe_bid = 0
        for each card in player table (reverse):
            if card is 'rose':
                increment safe_bid
            else:
                break
        safe_bid = safe_bid + 1
        if safe_bid < max_bid:
            return safe_bid
    return action
```

Listing 2: Python function responsible for bluffing. Translated to pseudocode.

The 'consider_bluff' function (Listing 2) is responsible for evaluating if a player is supposed to bluff during a specific round (by confirming the identity of the player, taking into account the bluff chance and making sure bluffing is only allowed when the last placed card is a Skull) and applying the actual bluff mechanics. If the player is indeed supposed to bluff then the maximum safe bid is calculated by counting all the roses found before encountering a Skull, in the reversed player stack. In the implementation used for the final experiment, this part of the code is redundant because the maximum safe bid will always be 0 (due to the condition requiring the Skull to always be on top). The next part of the code is responsible for increasing the maximum safe bid by 1, therefore initiating a bluff. Lastly, there is a check that makes sure the current bluffing bid cannot be equal or higher than the maximum allowed bid (which is the current number of cards on the table).

2.3 Experimental Setup

Five experimental scenarios were constructed to test the effectiveness of different strategies:

- **Scenario 1 (Random vs Random):**
  Both players take random actions in phase 1 and phase 2 of the game
- **Scenario 2 (Random vs Q-Learning):**
  One player takes random actions in phase 2 of the game while the other uses Q-learning. Phase 1 of the game is still randomized for both players.
- **Scenario 3 (Q-Learning vs Q-Learning and Bluffing):**
  Both players use Q-learning, but Player 2 is forced to bluff with a certain probability (only when the last placed card is a "Skull"). Phase 1 is randomized for both players.
- **Scenario 4 (Q-Learning vs Q-Learning and Bluffing Forbidden):**
  Both players use Q-learning, but one is forbidden from bluffing. Phase 1 is randomized for both players.
- **Scenario 5 (Q-Learning and Bluffing Forbidden vs Q-Learning and Bluffing Forbidden):**
  Both players use Q-learning, but bluffing is forbidden for both. Phase 1 is randomized for both players.

In all scenarios, the first phase (card placing phase) was handled by allowing each player to randomly place one card at a time until all cards are placed or until a player decides to move on to the second phase of the game by placing a bet, therefore the decision to move to the bidding phase could occur randomly as long as each player has placed at least one card in their stack. This simulates the uncertainty and variation that is expected in a real life game of Skull and Roses. The third phase (card reveal) was handled by

having the 'challenger' (the player that won the bidding phase) flip the number of cards they bid, starting by first revealing cards from their own stack and then the opponent's, if necessary.

In the scenarios where bluffing is forbidden, the code presented in Listing 3 will check if the bid value dictated by the Q-Table is higher than the maximum safe bid. If that is the case, then the value from the QLearning algorithm will be replaced by the maximum safe bid value, otherwise, the value from the Q-Table will be used. If the maximum safe bid value is lower or equal to the opponent's current bid, then the player will pass. In scenarios where bluffing was forbidden, the agent's action space was reduced by excluding bluffing actions. These constraints on bluffing reduced the available actions for the QLearning agent, potentially leading to the agent learning the same strategies.

```
    if self.forbid_bluffing:
        for card in reversed(self.table):
            safe_bid = 0
            if card == 'rose':
                safe_bid += 1
            else:
                break
        if safe_bid > action:
            return safe_bid
        else:
            return action
```

Listing 3: Code excerpt responsible for handling forbidding bluffing (simplified version).

For each scenario, the experiment ran for 1000 game rounds, repeated over 100 simulations. The total number of wins for each player was recorded and averaged across runs to ensure stable results.

- Epochs: Each simulation run consists of 1000 game rounds.
- Results: The number of wins for each player is recorded and averaged across runs to determine the effectiveness of the strategies.

The results of the simulations were exported to a CSV file for further analysis. The CSV file includes the total and average number of wins for each player in each scenario, but also a more detailed look at the running total for the last simulation, providing a comprehensive view of performance differences. Lastly, debugging statements have been implemented in the code, providing a log for the actions taken by both players in every round of the game. Upon manually examining this log, a viewer could verify the integrity of the gameplay (for example if all the rules are being followed).

3 Results

Although the experiment will be conducted for 100 simulations of 1000 rounds each, this is only because after this point, no differences can be observed in the results when increasing the number of simulations and epochs. Results for all five scenarios have been obtained and will be discussed further:

3.1 Scenario 1: Random vs Random

The first scenario revolves around two players that both use a random strategy to play the game. Therefore, there is no intelligent decision-making process in this scenario. It is meant to serve as a control scenario, as well as a debugging tool. By analyzing the results of this scenario we can draw conclusions with regards to how well the game rules have been implemented and making sure there is no unfair advantage for either player. The action that both players take for phase 1 and phase 2 of the game are dictated by a random function.

- Average Results for Scenario Random:
  Player 1 - 500.42 — Player 2 - 499.58

The results suggest that no player has an advantage, the average wins across 100x1000 rounds being 500.42 vs 499.58, which is almost a 50-50 split. Therefore, this scenario will do a very good job of serving as a control scenario for all the following results. Figure 3.1 offers a visual depiction of how the two players follow the same upward trend, ending up with very similar (almost equal after averaging) results.

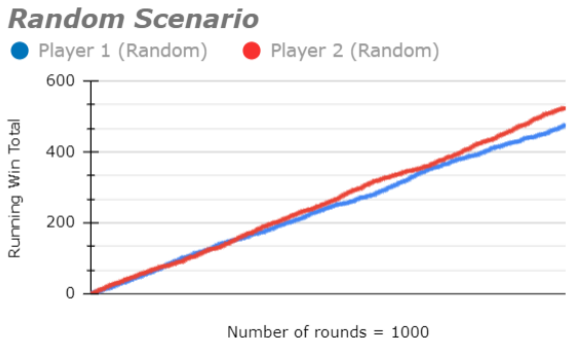

Figure 3.1: Running Win Total for 1 simulation in the Random Scenario

3.2 Scenario 2: Random vs Reinforcement Learning (QLearning)

This scenario is used to check the implementation of the reinforcement learning algorithm. In this case, Player 1 has a purely random strategy (the same as in scenario 1) while Player 2 employs a Q-Learning algorithm with an Epsilon Greedy policy. The parameters used by the reinforcement learning algorithm are as follows:

- self.alpha = 0.1
- self.gamma = 0.9
- epsilon = 0.2
- epsilon_decay = 0.995
- min_epsilon = 0.01
- FIXED_STARTER = 0
- ALL_CARDS_BEFORE_BID = 0

After running the test for 100x1000 rounds, these are the average results:

- Average Results for Scenario QL:
  Player 1 - 265.82 — Player 2 - 734.18

Clearly, the reinforcement learning agent shows a considerable advantage (approximately 26-74 split) over the player that relies on a random function for its actions. This is because Player 2 is able to find good strategies and exploit them. Figure 3.2 showcases the immediate performance advantage for the Q-Learning agent and the constant increase in performance across the epochs.

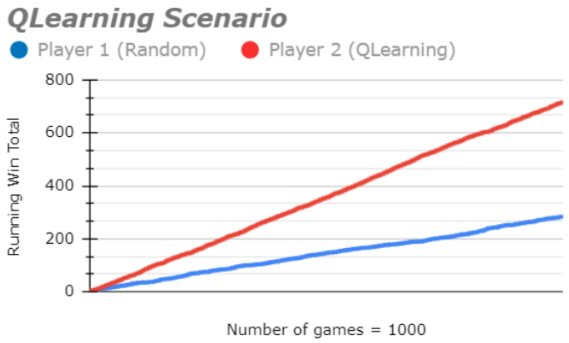

Figure 3.2: Running Win Total for 1 simulation in the QLearning Scenario

3.3 Scenario 3: Q-Learning vs QL & Bluffing

This scenario will test the efficiency of Bluffing (player 2) against a regular Q-Learning agent (player 1). If there is a statistically significant advantage for the bluffing player, the hypothesis will be confirmed. The parameters being used are the same as in scenario 2 with the addition of the following:

- USE_RANDOM_BLUFF_CHANCE = False
- BLUFF_CHANCE = 0.6

The bluff chance indicates that bluffing will occur for 60% of the time, only when the last card for player 2 is a Skull. After testing, the last card card is a Skull for player to in roughly 25% of the cases (results after 100x1000 rounds - Average skull as last card for Player 2: 250.96). Knowing this, we can calculate exactly how often bluffing will be used overall across all 100x1000 rounds of the game: 60% x 25% = 15%.

- Average Results for Scenario Bluffing:
  Player 1 - 494.78 — Player 2 - 505.22

The results show that the bluffing agent has a slight advantage over the non-bluffing agent.

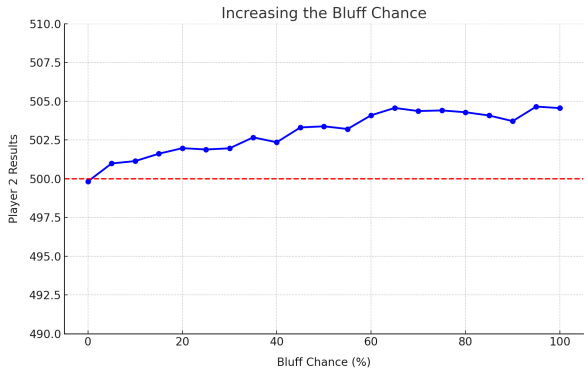

Figure 3.3: Increasing the BLUFF_CHANCE parameter by 0.05 in the 0-1 interval

The graph in Figure 3.3 shows the effects of gradually increasing the bluff_chance value from 0% to 100% (100% translating to a 25% bluffing integration across all games). When increasing the bluffing chance, we can see a gradual increase in performance for Player 2, however, around the 65% mark, a plateau is starting to form all the way to 100% bluffing chance. Unfortunately, the current implementation does not allow for further analysis (due to the rule stating that the skull needs to always be on top for bluffing to occur).

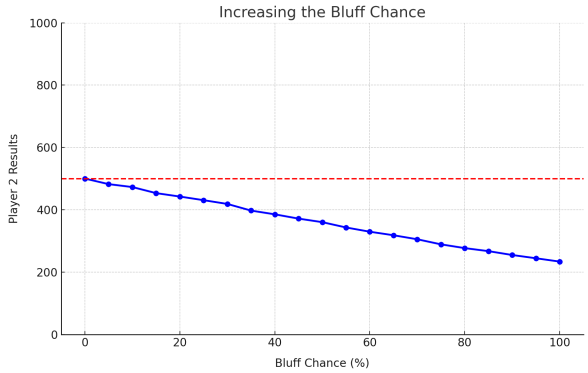

Figure 3.4: Increasing the BLUFF CHANCE parameter by 0.05 in the 0-1 interval without the Skull on top rule

Figure 3.4 showcases the performance of the bluffing agent when bluffing is allowed to happen solely on the bluff chance, not taking into account the placement of the cards during phase 1. The graph shows a downward trend in performance when increasing the bluffing frequency. This finding is to be expected because the more often a player bluffs, the more chances the other player (regular reinforcement learning) has to identify the bluffing pattern and not fall in the trap. Furthermore, it is less likely that the non-bluffing player can even increase the bid safely, making it more likely that the bluffing player is challenged.

3.4 Scenario 4 & Scenario 5

The last two scenarios involved forbidding the use of bluffing first just for player 2 and then for both players, respectively. The results are as follows:

- Average Results for Scenario QL Bluffing Forbidden (Player 2):
  Player 1 - 499.92 — Player 2 - 500.08

- Average Results for Scenario QL Bluffing Forbidden Both:
  Player 1 - 499.87 — Player 2 - 500.13

For both scenarios, no difference can be observed between the two players, indicating that not a lot of 'accidental' bluffing is occurring in the current implementation. This suggests that the QLearning algorithm prefers staying under the maximum safe bid.

3.5 Statistical Analysis

In order to confirm or reject the hypothesis, a statistical significance test has been conducted on the results obtained in the third scenario. The data used for the test is composed of a list of total won rounds by each player, for every simulation. In this study, a total of 100 simulations have been conducted, therefore, a total of 100 data points are used for each player. After conducting a Shapiro-Wilk test (P-Value = approximately 0.47, which is greater than the 0.05 threshold), we can conclude that the data obtained from Scenario 3 is approximately normally distributed (this can also be observed in Figure 3.5), therefore a Paired T-Test is suitable in order to test for statistical significance.

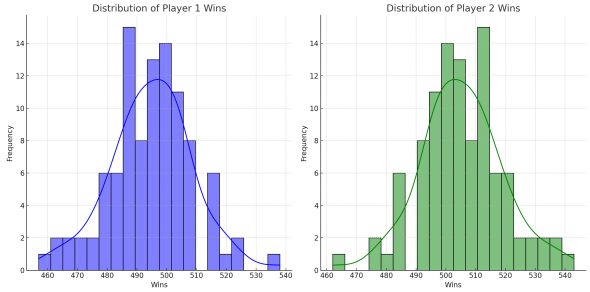

Figure 3.5: Distribution of wins for Player 1 and Player 2 for Scenario 3

- **Paired T-Test**
  After conducting a paired t-test (calculating the differences, obtaining the mean of the differences, calculating the standard deviation of the differences, obtaining the t-statistic, calculating the degrees of freedom and ultimately determining the p-value), the **p-value obtained is p = 0.00028**, well below the 0.05 significance threshold.

4 Discussion

4.1 Conclusion

The research was aimed at understanding how a bluffing strategy affects the performance of a player in the game of Skull & Roses. The results revealed that indeed there is a performance increase for the players that are using bluffing when compared to players that do not make use of bluffing. However, this boost in performance can only be observed under specific card layouts, indicating a very close relation between the first phase of the game (when the players place the cards on the table) and the second phase of the game (when the bidding war is waged). Conducting a statistical significance test (Paired T-Test) on the results obtained in Scenario 3 showed statistical significance (the obtained p-value was smaller than p=0.05). This leads to the conclusion that the results fail to reject the hypothesis. Therefore, bluffing is indeed increasing the performance of the player if the right game environment is provided.

4.2 Discussion & Literature

The research conducted by Friedman (1971) on optimal bluffing strategies in poker offers a theoretical foundation for the conditions under which bluffing can be effective in imperfect information games. Friedman employs game theory in order to figure out what is the frequency and optimal conditions in order to obtain maximum gains from bluffing in the game of Poker. Apart from figuring out if bluffing offers a competitive advantage in the game of Skull, this study parallels Friedman's because it showcased that bluffing offers an advantage only under specific card layouts. Both studies emphasize the role of deception in games of imperfect information. The big difference comes from the fact that Friedman's work predates current Reinforcement Learning and computing concepts and relied solely on mathematical constructs in order to reach a conclusion. Therefore, this study could be considered a practical demonstration of some of Friedman's conclusions.

Additionally, this study examines bluffing in the context of opponents that are learning over time. Unlike Friedman's static theoretical approach, this research incorporates dynamic interactions where the opponent could theoretically develop counter-bluff strategies. However, the results did not indicate that the opponent learned to counteract bluffing within the 1000 rounds time frame, suggesting that either the duration was too short for such strategies to emerge or that the learning algorithm requires further alterations and experimentation. This insight highlights the complexity of learning effective bluffing and counter-bluffing strategies in a relatively short period.

Another closely related study has been conducted by Marwala & Hurwitz (2009), focusing on the use of multi-agent systems and reinforcement learning to train agents on executing bluffing strategies. This research makes use of Temporal Difference Reinforcement Learning to allow agents to predict opponent's reactions based on both the actions of other players as well as their own cards. The findings demonstrate that agents can learn to bluff effectively and even recognise each other's bluffs, supporting the idea that bluffing can be optimized. This leads me to believe that further

research is needed on implementing bluffing in the game of Skull & Roses. I theorize that if Reinforcement Learning would be applied to all three phases of the game (alongside including more than two players in the game), the system would potentially be able to create its own bluffing strategies as well as recognizing other's bluffs. This claim is supported by the research conducted by Tuyls & Nowé (2005) on how Reinforcement Learning could be applied to train agents in reaching human levels of performance in strategic games. This could include human traits, such as recognizing bluffs. While the current study did not show agents learning counter-bluff strategies, humans, in contrast, explicitly attempt to determine whether others are bluffing through pattern recognition and psychological cues. However, there is no indication that a Reinforcement Learning implementation couldn't develop a human-like strategic approach that includes bluff detection, therefore further research might unveil such progress.

The findings discussed in this research could serve as a building block towards a more nuanced understanding of how bluffing influences various real life systems. For example, bluffing is a common strategy in high-stakes environments such as business negotiations. Understanding and optimizing bluffing strategies through reinforcement learning can help individuals and organizations develop more effective negotiation tactics, leading to better outcomes in competitive scenarios. Another example could be in the realm of cyber-security and fraud detection where bluffing can serve as a parallel to the deceptive tactics used by attackers. By training AI to recognize and respond to these strategies, we can enhance the effectiveness of systems designed to detect and prevent fraudulent activities.

The limitations faced by this study varied, however the main limitations were related to the resource intensive nature (both time-wise and computationally) of implementing Reinforcement Learning for the entirety of the game of Skull and Roses. This is why early on in the development process, a decision was made to focus only on the second phase of the game. Another decision that simplified the development process was the choice of using only a two-player configuration (the maximum number of players for Skull and Roses is 6). Implementing a higher player count could considerably help with the learning process because of the increased variety of strategies that are being employed simultaneously.

Further research could also focus on a different implementation of bluffing, such as allowing bluffing to emerge as a beneficial strategy on its own instead of forcing it. This could potentially be achieved by implementing a complex reward structure that dynamically provides slightly higher rewards for newly discovered strategies. The longer that strategy would be used, the lower the reward will drop (until reaching a default minimum), therefore incentivizing the exploration of other possible strategies, including bluffing and bluffing variations.

References

Buşoniu, L., Babuška, R., & De Schutter, B. (2008, March). A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *38*(2), 156–172. doi: 10.1109/TSMCC.2007.913919

Campbell, M., Hoane, A. J., & hsiung Hsu, F. (2002). Deep blue. *Artif. Intell.*, *134*, 57-83. doi: 10.1016/S0004-3702(01)00129-1

Dufwenberg, M. (2011). Game theory. *Wiley interdisciplinary reviews. Cognitive science*, *2 2*, 167-173. doi: 10.1002/wcs.119

Fares Fourati, M.-S. A., Vaneet Aggarwal. (2024). Stochastic q-learning for large discrete action spaces. *arXiv preprint arXiv:2405.10310*.

Friedman, L. (1971). Optimal bluffing strategies in poker. *Management Science*, *17*, 764-771. doi: 10.1287/MNSC.17.12.B764

Guazzini, A., & Vilone, D. (2013). Bluffing as a rational strategy in a simple poker-like game model. *Journal of Complex Systems*, *2013*, 1-6. doi: 10.1155/2013/390454

Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-learning algorithms: A comprehensive classification and applications. *IEEE*

Access, *7*, 133653-133667. doi: 10.1109/ACCESS.2019.2941229

Kaelbling, L., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *J. Artif. Intell. Res.*, *4*, 237-285. doi: 10.1613/jair.301

Marly, H. (2011). Skull game. *Lui-même*.

Marwala, T., & Hurwitz, E. (2009). A multi-agent approach to bluffing. *Multiagent Systems*. doi: 10.5772/6603

Schofield, M., & Thielscher, M. (2019). General game playing with imperfect information. *J. Artif. Intell. Res.*, *66*, 901-935. doi: 10.1613/jair.1.11844

Sewak, M. (2019). Q-learning in code. *Deep Reinforcement Learning*. doi: 10.1007/978-981-13-8285-7

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, *550*, 354-359. doi: 10.1038/nature24270

Syafiie, S., Tadeo, F., & Martinez, E. (2004). Softmax and -greedy policies applied to process control. *IFAC Proceedings Volumes*, *37*, 729-734. doi: 10.1016/S1474-6670(17)31556-2

Tuyls, K., & Nowé, A. (2005). Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review*, *20*, 63 - 90. doi: 10.1017/S026988890500041X

Ye, W., & Chen, D. (2022). Analysis of performance measure in q learning with ucb exploration. *Mathematics*. doi: 10.3390/math10040575

Yoshida, W., Dolan, R., & Friston, K. J. (2008). Game theory of mind. *PLoS Computational Biology*, *4*. doi: 10.1371/journal.pcbi.1000254