Universiteit Leiden
The Netherlands

Opleiding Informatica

Creating Agents for
the Card Game Rikken

Mika Bakker

Supervisors:
Dr. E.P.L. van Nieuwenburg & Dr. J.N. van Rijn

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

10/7/2024

Abstract

This thesis looks at the game Rikken. Rikken is a card game with strategy, randomness, and imperfect information. This makes it a compelling game for testing different algorithms. In this thesis, we look at several agents created for the game of Rikken. Additionally, we experiment with different combinations of these agents, namely, a random agent, a greedy agent, and a Monte Carlo agent. Lastly, we look at more detailed results of the experiments, to get a better understanding of the strategy incorporated by the Monte Carlo agent. Our results include a Monte Carlo agent that beats the greedy agent, winning 69.3% of games in one experiment. We discuss the differences in strategies that lead to that result.

Contents

1 Introduction

Card games form an interesting research topic in the field of artificial intelligence (AI). In many card games, players do not have full information about the cards held by the other players, meaning they must deal with imperfect information. For an AI agent, this lack of information is what makes the problem challenging. An AI agent will have to consider all possible scenarios or involve heuristics and estimates to make decisions. For computationally complex games, considering all possible scenarios is infeasible. As a result, the usage of apt heuristics and estimations becomes more important.

In this thesis, we consider a trick-taking card game called Rikken. This game belongs to the same genre of games as Bridge, Klaverjas, and Hearts. Trick-taking card games are a popular genre of card games centered around *tricks*. Tricks are rounds where every player plays one card, the player that played the highest valued card wins said trick. Whether winning a trick results in a positive score can differ from game to game, and sometimes even from trick to trick. Trick-taking games require strategy and planning, in order to navigate decisions with limited information. This makes them an appealing but challenging topic of artificial intelligence research.

The main focus of this project is the implementation of a Monte Carlo agent. By definition, the Monte Carlo method [MU49] makes use of random sampling. Monte Carlo algorithms can be used to calculate the expected outcome of games. This is useful when an exhaustive search of the game tree is not possible, which is the case for Rikken. For this project, a version of the game is implemented, alongside three different agents to play the game. Specifically, the agents are a random agent, a greedy agent, and a Monte Carlo agent. The relative strength of these agents can be compared by having them play against each other. We expect the Monte Carlo agent to be able to play our version of the game Rikken better than the random agent and the greedy agent. The main question is, how much does a Monte Carlo agent improve the performance over a greedy and random agent?

As a secondary objective, we aim to determine the strategy of the Monte Carlo agent. However, determining the exact strategy of the Monte Carlo agent is challenging. The state space of Rikken is large, and the decision-making process of the Monte Carlo agent is non-deterministic. Nevertheless, we can analyse a large number of games, to get an understanding of the playstyle of the Monte Carlo agent. On top of that, we will highlight the first action of the Monte Carlo player specifically. Together, we can use this information to create rules of thumb. From that, we could form a basic strategy, which is understandable to a human player.

Various strategies exist for agents to play trick-taking card games, and we will explore some of them in this thesis. In Chapter 2, we will first introduce our implementation of Rikken and the simplifications we have had to make. Then in Chapter 3, we will discuss related work, such as research that has been done on other trick-taking games and how it connects to our work. In Chapter 4, we will elaborate on the different agents that were used and their implementation. After that, Chapter 5 displays the results of our experiments, showing the effect of using different agents, as well as looking at some of their strategies. Lastly, in Chapter 6 we discuss what conclusions can be drawn from said experiments. On top of that, we explore interesting options to continue research on the game of Rikken, and possible improvements on the agents created for this thesis.

2 Rules of Rikken

In this section, I will explain the rules of Rikken. Some simplifications have been made, which allow us to focus more on specific aspects of the game. These simplifications will be explained later.

Rikken is played with a standard deck without jokers, for a total of 52 cards. Where 2 is the lowest value, and Ace is the highest. It is played with four players. Interestingly, the cards are not shuffled in between rounds. Commonly, the deck is only cut once. At the start of each round, each player is dealt 13 cards. Dealing is supposed to be done according to a specific pattern. Commonly used examples of these patterns are 6-7 and 4-5-4. 6-7 means player 1 is dealt six cards from the top of the pile, then player 2 is dealt six cards, etc. After all players have received six cards, all players are dealt seven cards in the same way, starting from player 1. The combination of not shuffling the cards and dealing in these specific patterns has a large impact on the game. Generally, it makes players' hands more likely to have many cards of the same suit (and few or none of the other suits). However, the experiments in this thesis are done using randomly distributed cards.

2.1 Bidding

After the cards have been dealt, the bidding round starts. Players bet on the number of tricks they expect to win. This can either be with the help of a "maat"(mate), or alone. Where a bid of n tricks alone is valued higher than a bid of n tricks with a mate, but lower than any bid of $n + 1$ tricks. The minimum starting bid is called "Rik", which means the player will aim to win at least 8 tricks, with the help of a mate.
Our implementation of the game foregoes the bidding round completely. This was done to reduce the complexity of the game, as including a bidding round would require implementing a separate bidding algorithm. This allowed us to focus more on the trick-taking aspect of the game. In this thesis, all games are played with the first player playing for the standard contract "Rik". This player is called the "Rikker". This means that in the context of our experiments, the Rikker's goal is always to win at least eight tricks, with the help of a mate.

2.2 After Bidding

After what would normally be a bidding round, the Rikker announces the trump suit for that round. If the contract includes that the Rikker plays with a mate, the Rikker will then pick a mate. This is done by calling out an Ace, which is not in their own hand. Whoever is holding that Ace will be the mate for that round. This means nobody except the mate themselves knows who is playing as the mate that round until they play that Ace. Most rulesets include a special kind of contract that must be played if one player holds all four aces. In that case, the Rikker can choose a king as their mate. The experiments in this thesis instead have either a set or a random mate in all cases.

2.3 Gameplay

After the trump suit and mate have been decided, the playing phase can start. In this implementation of the game, the Rikker is the first to act in the first round. This phase ends when all players are out of cards. This means the playing phase consists of 13 tricks.

2.3.1 Following Suit

The players take turns in a clockwise direction. After the player that is first to act at any trick has played a card from their hand, all remaining players are required to "follow suit". This means that they are forced to play a card of the same suit as the first player (the "asked suit") if they are holding any. If a player is unable to follow suit, they may play any card from their hand. This process repeats itself in the next trick, possibly with a different asked suit.

2.3.2 Winning a Trick

In general, the player that played the highest card of the asked suit wins the trick. The exception is when trump cards are played. A trump card always wins over any non-trump card. If multiple trump cards have been played, the player that played the highest trump card wins the trick. Whoever wins the trick, is first to act during the next trick.

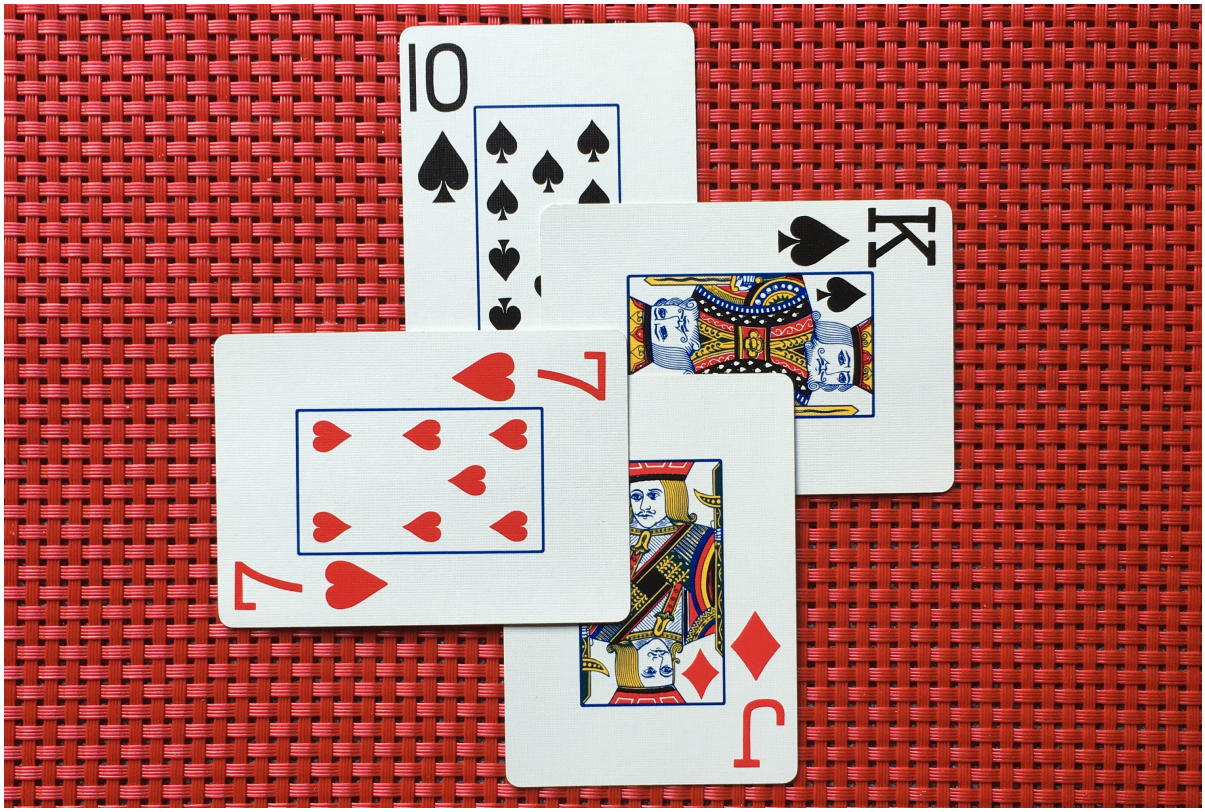

Figure 1: Example of a single trick

Figure 1 shows what a trick could look like. Let's assume North is the first to act and plays the 10♠. East is next and plays the K♠, as they are required to follow suit. South does not have any spades. Therefore, they can play any card they want. South plays the J♦. West does not have spades either and plays the 7♥. If we ignore trump cards (say, clubs are the trump suit), East wins this trick by playing the highest card of suit North opened with (spades). Alternatively, if hearts are the trump suit, West would win this trick. Despite playing the lowest card, the 7♥ wins, as trump cards are always ranked higher than non-trump cards.

2.4 Winning the game

Rikken has a somewhat unusual scoring system. It is a "zero-sum game", which means the combined score of all players is always equal to zero. After each round, losing players pay points to the winning players. The amount depends on the contract and the amount of tricks won. Instead of keeping track of the exact number of points each player has, we decided to only look at whether the contract was completed or not. This allows us to compare win rates in different scenarios. In any round where the Rikker and the mate win a combined number of tricks $t \geq 8$, the contract is completed, and the round is classified as a win for the Rikker.

2.5 Additional rules

There are a few specific rules that I did not include in my implementation of the game. According to some rule books [regb], the mate is required to play the ace, which was called by the Rikker before the start of the round, the first time they are required to play a card of that suit. Thereby revealing their identity as the mate. In other variations [rega], the mate is only required to play their ace if the Rikker asks them to, which is only possible when the Rikker is first to act. This is a special turn, where the other players are not allowed to trump in.
Another rule that is sometimes used is forced trumping. In the scenario where a trump card has been played, other players who cannot follow suit are required to play a trump card as well (if they have any). Despite the trump suit not being the suit they were originally asked to follow.

3 Related Work

In order to find work relevant to our study on the game of Rikken, we have to expand our view to the field of trick-taking games. Rikken is a relatively local and unknown game, which makes it nearly impossible to find relevant work on this specific game. However, plenty of research has been done on games similar to Rikken. Here are a few examples of other trick-taking card games that have already been topics of research and why they are of interest to us:

- Bridge: Bridge is a classic trick-taking card game and the topic of many studies. Giving us a great source of findings and ideas that could apply to Rikken as well. An important difference between Bridge and Rikken is that Bridge is played by two teams of two, rather than four individual players and teams changing each round. This means that both teams can form specific strategies before the game. On top of that, there is a "dummy" player that plays with their cards visible to all other players and whose actions are determined by their teammate. Applying a Monte Carlo algorithm to Rikken or Bridge is quite similar. In both cases, there is the problem of imperfect information. A strong Bridge agent has been created by ML Ginsberg [Gin01]. This paper discusses a way for Monte Carlo to approach this challenge of imperfect information in card games, similar to the method used for this thesis and described in Section 4.3.2. As well as pointing out some of its weaknesses.

- Skat: Instead of randomly distributing cards, we can look at more clever ways to generate playouts with hidden information. Kumpferschmid and Helmert already applied Monte Carlo methods to the game Skat [KH06] in a different way. Skat is the national game of Germany. Despite being a trick-taking game, Skat differs from Rikken in many ways. Skat is played by only three players, using only 32 cards. A notable similarity to Rikken is that the teams are dynamic. Each round has different players playing as a team, which is determined during the bidding phase. The article elaborates further on the challenges of using Monte Carlo in trick-taking card games. And points out that assigning cards to other players can be improved by taking into account the conditional probability of having specific cards. This means that some card configurations should be more likely than others, in order to create playouts that favor more realistic scenarios. This idea has not been explored in the experiments that will be discussed later, but could be tested as a possible improvement on the methods described in this thesis.

- Klaverjas: Klaverjas has become quite a popular research topic among computer scientists at Leiden University, which served as a source of inspiration for this thesis. Klaverjas is another Dutch trick-taking card game. It has been studied by e.g. Van Rijn, Takes, and Vis [vRTV18]. As well as being the topic of multiple theses such as [Hor22], [Zwe20] and [Hoo17]. The latter of which also served as an inspiration for the implementation of the Monte Carlo agent. Specifically how playouts can be generated with constraints on other player's cards. The stated method was used in experiments for this thesis, which is described in more detail in Section 4.3.2.

4 Agents

The experiments feature three different agents in various configurations. Each of these agents follows an entirely different decision-making process. This section explains how each of the different agents was implemented.

4.1 Random Agent

The first agent created is the random agent. This player simply selects a random card from its hand. The random player does need to play according to the rules and follow suit. If playing the selected card is a legal action, the card is played. If playing the selected card is illegal, the player keeps selecting random cards until it finds one that is a legal play.
The random agent is useful as an agent that we can compare the strength of other agents to. It is also useful for experiments as it uses very little computation time.

4.2 Greedy Agent

The greedy agent uses a very simple greedy strategy. Whenever it has the opportunity to win a trick, it will try to do so. The greedy agent's decision process is as follows:
First, it checks whether it is first to act. If the greedy player is first to act, it simply plays the highest card in its hand. (In the case of a tie, the first highest card encountered when iterating over the hand is played). The agent does not value the trump suit any differently than the other suits in this case.
If the greedy player is not the first to act, it looks whether any of the cards they are holding can legally win the trick. If they can win the trick, they will play the **lowest** card available that beats the cards that are already on the table.
If the greedy player does not have a card that can win the current trick, it simply plays the lowest card it can legally play.

4.3 Monte Carlo Agent

The third agent uses a Monte Carlo algorithm to determine which card to play. This agent performs many random playouts and keeps track of the results. A playout is defined as a simulation of the game using random moves for each player, starting from the current state of the game and ending when all players are out of cards. The MC agent has a counter for each card in their hand. Before each playout, it selects a random card from its hand, which is played first. Afterwards, the result of this playout is added to the counter of the selected card. The result is either 1 or 0 for a win or loss respectively. Then, the card that results in the most won playouts is played in the "real" game. The number of playouts P depends on the number of cards left in the player's hand (c), and a parameter n. The value of P is determined as follows:

$$P = n * c^2 \tag{1}$$

The value of parameter n remains constant throughout the game. The experiment in section 5.1 shows how adjusting this parameter influences the win rate of the Monte Carlo player in a game with random agents. In all other experiments, $n = 8$ is used.

The value of c is defined as the number of cards left in the agent's hand. If we assume $n = 8$, this results in a total of $P = 8 * 13^2 = 1,352$ playouts in the first turn. On the other hand, only $P = 8 * 1^2 = 8$ playouts are performed for the last turn. It makes sense to have a variable number of playouts based on the number of cards left, as the search space in the first turn is orders of magnitude larger than later in the game. Moreover, eight playouts are more than necessary for the last turn, as the player only has one card left to play, which makes any playouts pointless. However, the time spent on these eight short playouts is negligible in comparison to the thousands of playouts done in earlier stages of the game.
To generate a random playout, the Monte Carlo agent creates a copy of the current state of the game. However, we cannot let the agent create an exact copy of the current game, as it does not have knowledge of the other players' hands. So, in order to perform a random playout without giving the MC player an unfair advantage, the agent itself assigns plausible cards to its opponents (while keeping its own hand intact). In this context, plausible cards for each opponent can be determined by keeping track of previous tricks. Certain cards can be ruled out after a player is not able to follow suit. This is explained in more detail in section 4.3.2.

4.3.1 Playout evaluation

At the end of each playout, the state of the game needs to be evaluated. This evaluation is necessary to decide which card is the correct or preferred action. The evaluation function used gives a playout where the Rikker and mate together have won at least eight tricks a score of 1. A playout where the Rikker and mate win less than eight combined tricks gets a score of 0. Technically, this uses information which is not always available to all players. Namely, the identity of the mate is only confirmed once they play their ace. Before that, the other players can only guess who the mate is. This means that the evaluation of a playout is not completely fair. However, in most real games, the identity of the mate is revealed very early in the game. Additionally, this implementation of Rikken does not include the component where the mate is selected by choosing an ace. That means there is no play where the identity of the mate is revealed to the other players. Therefore, giving the Monte Carlo player information on which player is the mate makes it easy to evaluate playouts in a way that should lead to realistic results.

4.3.2 Dealing for playouts

There are multiple ways to go about redistributing the cards for random playouts. As the first step, my implementation creates a pile of all the cards the Monte Carlo player has not yet "seen". This pile contains all cards that have not yet been played and are not in the MC player's hand. Which is equal to all the cards in the hands of all opponents combined. At this point, one could decide to simply shuffle the pile of remaining cards and deal them to the opposing players.
However, as Hoogenboom [Hoo17] points out, there is information to take into account about which cards other players can and cannot hold. Whenever a player is unable to follow suit, one can know for certain that this player cannot hold any more cards of that specific suit. As a result, simply giving opponents random cards will regularly create playouts in situations that are impossible from the MC player's perspective, which can be considered less valuable than plausible playouts.
Therefore, I implemented a system similar to the one described by Hoogenboom [Hoo17], which keeps track of which suit a player can and cannot hold in a matrix of booleans, shown in Figure 2:

	Player 0	Player 1	Player 2	Player 3
♠	false	false	true	false
♡	false	true	true	false
♣	false	false	false	false
♢	true	true	false	true

Figure 2: Example of a matrix by Hoogenboom [Hoo17] that shows whether a player can be holding cards of a given suit or not

Now, after dealing new cards to the other players for a playout, we can check whether it creates a plausible playout using a matrix like that in Figure 2. If a player is dealt a card it cannot have, the deck is reshuffled and dealt again until a realistic card configuration is found. This does have an additional performance cost. There are situations where this process can take a significant amount of time. An alternative would be to deal the cards per suit, where cards of any given suit are dealt exclusively to the players that can hold cards of that suit, as this can save computation time. However, it is important to make sure the cards are still distributed in a uniformly random manner.

5 Experiments

5.1 Monte Carlo playouts

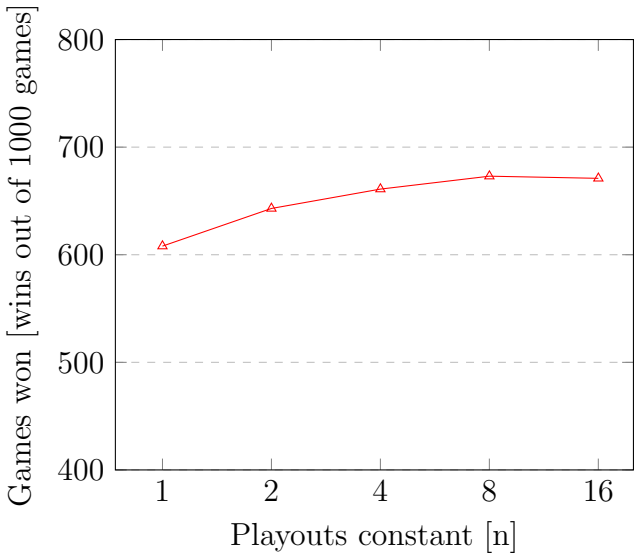

Figure 3: Number of games won out of 1000 by the Monte Carlo player as the Rikker, using $n * c^2$ playouts. While playing with a random agent as the mate, and two random agents as opponents

Figure 3 shows the results of an experiment to determine an optimal choice for n, see Eq. 1. The experiment involves one Monte Carlo agent as the Rikker and a random agent as the mate, playing against two random agent opponents. We can see how the number of playouts affects the win rate of the Monte Carlo agent. Based on the results of these experiments, we can assume that there is

very little improvement when using more than $8 * c^2$ playouts. In all the experiments described later in this thesis, all Monte Carlo agents used the value $n = 8$ when calculating the number of playouts for a given turn.
Another takeaway from Figure 3, is that it seems very difficult to win more than 70% of games when one's partner and opponents play randomly. In the next paragraph, we will see how different agent configurations influence the results.

5.2 Results using different agents

To get a good idea of the relative strength of our different agents, we need some understanding of the dynamics of the game. Especially because four players are competing in two teams (as well as small differences between playing as the Rikker or the Mate). Therefore, we will look at several experiments that include different combinations of agents.

Player types							
Rikker	Mate	Opp.1	Opp. 2	Games played	Wins	Win %	Std. deviation (%)
Random	Random	Random	Random	1,000,000	479,443	47.9	0.050
MC	Random	Random	Random	1,000	673	67.3	1.48
MC	MC	Random	Random	1,000	802	80.2	1.26
MC	Random	MC	Random	1,000	502	50.2	1.58
MC	MC	MC	MC	1,000	513	51.3	1.58
Greedy	Random	Random	Random	10,000	6,064	60.6	0.49
MC	Random	Greedy	Random	1,000	693	69.3	1.46
Greedy	MC	Random	Random	10,000	7,327	73.3	0.44
MC	MC	Greedy	Greedy	10,000	6,205	62.1	0.49

Table 1: Results of experiments using different combinations of agents. A game is won if the Rikker and mate complete the contract by winning eight or more tricks

Table 1 summarizes the main findings of this thesis. The left-hand side of Table 1 shows the different configurations of agents that were experimented with. The right-hand side of the table shows the results of the corresponding experiment. An interesting starting point in Table 1 is the games played where all agents use the same strategy (row 1 and row 4). In both of these experiments, the win rates are close to 50%. The same is the case for the experiment in row 3, where both teams also consist of the same type of agents: one MC agent and one random agent. Again, the resulting win rate is very close to 50%. This makes it seem like the game is at least somewhat fair and that before the cards are dealt, two teams using similar strategies have a roughly equal chance of winning. A game with four random agents appears to be slightly in the favor of the opposing players. However, adding more Monte Carlo players to each team appears to push the odds slightly into the Rikker's favor. Based on the results, the Monte Carlo player seems to be the strongest of the three agents. Achieving the highest win rate against two random agents (80%), as well as defeating two greedy opponents ~ 60% of the time.

5.3 Monte Carlo on the first turn

To gain some insight into the strategy and decision-making of the Monte Carlo player, I performed an experiment where we let the Monte Carlo agent play only the very first card. This means that the MC player has a full hand of 13 cards, and there are currently no cards on the table (the player is first to act in the first round). The reason we only look at the first turn is because the number of possible game states grows extremely rapidly once multiple players have acted. The number of possible plays on turn 1 is 52, as each card in the deck can be the first card played. We are exclusively looking at the card on the table. This means we can combine the results of equally valued cards that are not trump cards. When diamonds are the trump suit, there is no difference in value between a 7 of hearts and a 7 of spades. This symmetry means there are only $2 * 13 = 26$ unique first plays. Either the card is a trump card or not, and there are 13 possible values of cards. This does ignore the other cards in the MC player's hand. But there are $\binom{52}{13} = 635,013,559,600$ possible starting hands. Which makes defining the optimal play for every starting hand an astronomic task.

	Trump suit	Non-trump suit
2	2,873	4,285
3	2,618	4,295
4	2,391	4,187
5	2,251	3,975
6	2,149	3,851
7	2,162	3,854
8	2,120	3,975
9	2,067	4,032
10	2,371	4,366
J	2,767	4,691
Q	3,501	5,365
K	4,703	6,544
A	6,551	8,056
Total	38,524	61,476

Table 2: Cards played by a Monte Carlo agent in the first turn of the game, over 100,000 games. All three non-trump suits are added together

Based on Table 2, we can make a couple of observations about how the Monte Carlo agent plays the first trick. Firstly, the MC favours playing trump cards over non-trump cards. The ratio of the total number of trump cards to non-trump cards is $1 : 3$. When we look at the totals of cards that were played first at the bottom of Table 2, the ratio is roughly $1 : 1.6$. A reason for this could be that it forces all other players to play a trump card. This is positive because it lowers their ability to win tricks using trump cards in later stages of the game, although it does come at the cost of one of its own trump cards.
Secondly, playing a high card is the preferred option. In both the trump and the non-trump category, the Ace, King, and Queen are the most played cards (in that order). Seemingly, the Monte Carlo agent tends to try to win the first trick when they have a high probability of doing so. And even if the agent does not win the first trick, opening with a high card at least forces the opponent to also

play one of their strong cards to win that trick.
Somewhat surprisingly, the MC agent is more likely to start with a very low card (2 or 3) than other low to medium-high cards. This is especially noticeable in the games where the MC player played a trump card first. A possible explanation is that the MC player would rather play one of their weakest cards and concede the first trick than play a medium-strength card and probably lose the trick anyway. Because playing a weak card allows them to keep a better hand for later tricks. Playing a trump 2 or a trump 3 might be a strong move because it forces all other players to play their (stronger) trump cards, at a relatively low cost for the Monte Carlo player, as a 2 or a 3 is unlikely to win a trick anyway.

5.4 Looking at different strategies

To get a better understanding of how the games play out, we can also look at how often tricks are won by each player by round. In this paragraph, we will take a closer look at how the games from the experiments in row 8 and row 9 of Table 1 played out.

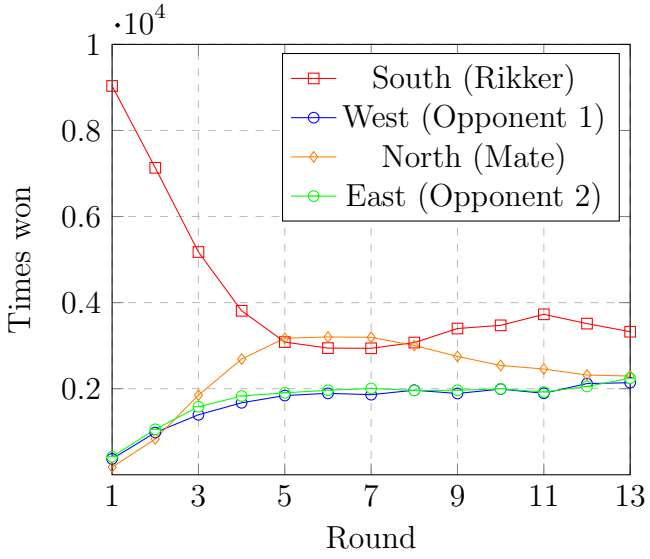

Figure 4: Number of tricks won by each player per round. The Rikker in red is a greedy agent, the mate shown in orange is a Monte Carlo agent. The opponents shown in green and blue are random agents.

Figure 4 shows which tricks were won by each player in the experiment from row 8 in Table 1. In this experiment, the Rikker (South) is a greedy agent. The mate (North) is a Monte Carlo agent. The two opponents (West and East) are random agents. The idea behind this experiment is to see if one greedy agent and one Monte Carlo agent would complement each other well when playing as a team. From Table 1, we learned that this combination of agents is reasonably strong (winning 73,3% of games), but not as strong as a team of two Monte Carlo agents.
The greedy Rikker always starts by playing their highest card. Figure 4 clearly shows this, with the red line winning roughly 90% of tricks in the first round. As well as continuing to win the most tricks up until round 5, where the Monte Carlo player starts winning slightly more. From round 8 up until round 11, the Rikker's win rate starts going up again, winning more than the Monte Carlo

agent. Presumably, this is the point where the players start to run out of cards from specific suits. Which allows them to start using trump cards to win tricks they could not win otherwise. The Rikker chooses the trump suit and picks the suit of which they are holding the most cards. So it makes sense that the Rikker starts winning more at this point in the game. Since the Rikker is a greedy agent, they will use their trump cards as soon as possible (except for the unlikely scenario where another player plays a higher trump card before them). A possible explanation for the dip in wins for the greedy player past the 11^{th} round is that they run out of trump cards at that point.

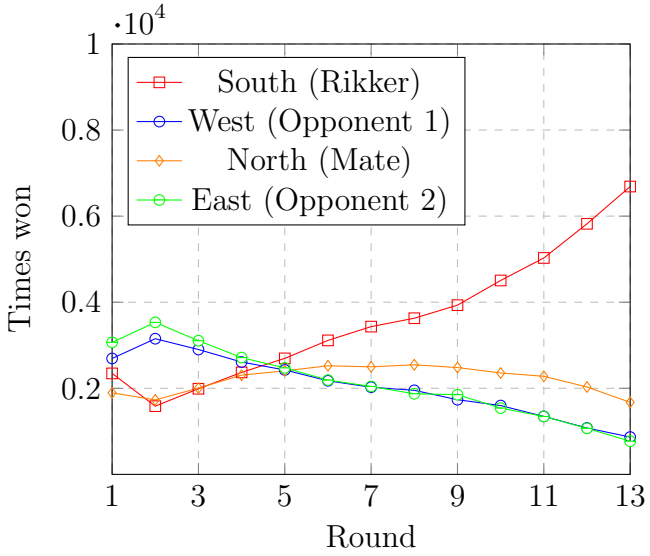

Figure 5: Number of tricks won by each player per round. The Rikker shown in red, and the Mate shown in orange are Monte Carlo agents. The opponents shown in blue and green are greedy agents.

Figure 5 shows the results of the experiment in row 9 of Table 1. In this experiment, two Monte Carlo agents (Rikker and Mate) play against two greedy agents. The two greedy opponents follow almost the same line, peaking in round two and only declining afterward. This is in line with what can be expected from a greedy agent. It tries to win any trick as soon as it gets the chance but runs out of steam fairly quickly. One possible explanation for the greedy agents not winning round 1 as often as they win round 2 is that the Monte Carlo Rikker employs a strategy called "troeven trekken" (pulling trump cards). This is where the Rikker opens with a trump card on the first turn (usually the ace). This forces all other players to play one of their trump cards, which greatly reduces their ability to trump during the later stages of the game. This opens up the opportunity for the Rikker to win more during the later rounds, as his trump cards will more often end up uncontested. The win rate of the Rikker in Figure 5 keeps increasing as the game goes on, up to almost 70%. The Monte Carlo Rikker is a lot better at using the advantage of having more trump cards in the endgame than the greedy agent, which had a peak at round 11 of just below 40%.
Another observation is that the other Monte Carlo player (the mate) has a fairly consistent win rate throughout the game of around 20-25%, which slowly declines towards the end. The mate knows that any tricks won by the Rikker also count towards themselves winning. So, instead of trying to win as many tricks as possible, the mate might intentionally play a weaker card of a suit the Rikker no longer has. As this allows the Rikker to trump in, making their advantage of having more trump cards even stronger.

6 Conclusions and Further Research

For this thesis, an implementation of the game Rikken was created. Additionally, we implemented three agents using different algorithms of varying complexity. The greedy agent does reasonably well against random players but does not fare well against the Monte Carlo agent. The highest achieved win rate is 80.2%, by a team of two Monte Carlo players against two random players. Although this is a reasonably high percentage, it means there is still room to improve as well. A team of two Monte Carlo agents achieved a win rate of 62.1% against a team of two greedy agents. Although the team of Monte Carlo agents had the advantage of going first, the result suggests that the Monte Carlo agent is stronger than the greedy agent. Based on the results in Table 1, we can conclude that the Monte Carlo agent came out as the strongest agent in these experiments.
Concerning the strategy adopted by the Monte Carlo agent, it appears that playing the ace of the trump suit is the preferred first move. This is not always an option. However, the Monte Carlo agent shows a preference for trump cards when starting in the first round. It is most likely to select a high card (Queen or higher). Interestingly, it elects to start with a low-value card more often than a medium-value card in round one.
In general, the Monte Carlo agent's strategy appears to be centered around winning the later rounds. More often than not, the Monte Carlo agent does not win many of the early rounds. Instead, it conserves its strong cards and uses those to win most of the tricks later in the game.

As for further research, various options are open to be explored when it comes to agents for the game of Rikken. One example is looking at the effect of using different shuffling and dealing methods, such as the dealing patterns described in Section 2.
Furthermore, many implementations of Monte Carlo agents different from the one described in Section 4.3 are possible. It could be worth looking into what effect a change in budget will have on the Monte Carlo agent's performance. On top of that, different ways of assigning cards to opponents for random playouts might also give interesting results. Assigning cards following the constraints described in Section 4.3.2 can take significant computation time. A different implementation, like dealing cards per suit or possibly even ignoring the matrix shown in Figure 2 can save large amounts of computation time, which could then be used to perform more playouts.
Another question is whether the use of Monte Carlo Tree Search [BPW+12] could make for a stronger agent. And whether it is better at navigating through this game, where most information is initially hidden but revealed throughout the round.
Once a better understanding of the playing phase is achieved, there is still a lot left to explore in the bidding phase of the game. This would require some ability to evaluate a player's starting hand. The bidding component of Rikken allows many options, which result in wildly different goals and games. Therefore, creating an extremely strong all-round agent for Rikken could prove to be an interesting but challenging task.

References

[BPW+12] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Gin01] M.L. Ginsberg. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:303–358, 2001.

[Hoo17] C. Hoogenboom. Using probabilities to enhance Monte Carlo search in the Dutch card game Klaverjas, 2017.

[Hor22] L. Hordijk. How to create an agent for the game of Klaverjas using Random Forests, 2022.

[KH06] S. Kupferschmid and M. Helmert. A Skat player based on Monte-Carlo simulation. In *International Conference on Computers and Games*, pages 135–147. Springer, 2006.

[MU49] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. PMID: 18139350.

[rega] Rikken. https://www.spelregels.eu/rikken. Accessed: 12-2-2024.

[regb] Spelregels Rikken. https://kaartspellen.online/kenniscentrum/spelregels/spelregels-rikken/. Accessed: 12-2-2024.

[vRTV18] J.N. van Rijn, F.W. Takes, and J.K. Vis. Computing and predicting winning hands in the trick-taking game of Klaverjas. In *Benelux Conference on Artificial Intelligence*, pages 106–120. Springer, 2018.

[Zwe20] M. Zwetsloot. Predicting the outcome of the card game Klaverjas, 2020.