Universiteit Leiden
The Netherlands

Opleiding Informatica

Agents for Kuhhandel

Marco van Dijk

Supervisors:
Walter Kosters
Jeannette de Graaf

BACHELOR THESIS

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

01/08/2018

Abstract

KUHHANDEL is a card game where three to five players compete for animal cards. Collecting four of a specific animal card results in completing a set, earning points for the player who collected them. Players can earn these animal cards by winning cattle trades against other players or by winning auction rounds. A key element of the game is bluffing. A cattle trade requires two players to place blind bids for an animal card and during auction rounds players are allowed to bid more money than they actually possess. For this thesis we implemented a simplified version of KUHHANDEL and we compare the performance of various agents created to play the game.

Contents

Chapter 1

Introduction

KUHHANDEL is a German game which was published in 1985 by Ravensburger. The game was also published in Dutch as *Koehandel*, in English as *You're Bluffing!*, in Portuguese as *Pague pra ver* and in French as *Boursicocotte*. Since some editions of the game follow slightly different rules, we will adhere to the rules of the Dutch edition of the game. It is played with three to five players and the goal of the game is to score as many points as possible by collecting sets of animals. Players are able to gather these animal cards by winning auctions or by trading with their opponents.

Figure 1.1: Picture of a game in progress. [1]

When players receive an animal card, they have to place it face up on the table in front of them. This way, everyone can see which animal cards belongs to whom. Money cards are held in the players hand and are thus unknown to the other players. This requires everyone to keep track of who receives what money cards in order to exploit this during a *cattle trade*, for example. The exact rules will be covered in Chapter 2.

1.1 Thesis overview

In this paper we will describe and compare three different agents we implemented which are able to play the game KUHHANDEL. The first agent plays the game randomly, the second agent determines the best move using the Monte Carlo algorithm, and the third agent plays based on some pre-defined rules.
We will also briefly compare two different methods of dealing with large move set sizes for our Monte Carlo agent.

In Chapter 2 we describe in detail the rules of the game. In Chapter 3 we discuss related work. Chapter 4 describes the implementation of our agents, which are evaluated in Chapter 5. Finally, we conclude in Chapter 6.

This paper was written as a bachelor thesis at the Leiden Institue of Advanced Computer Science of Leiden University, under supervision of Walter Kosters and Jeannette de Graaf.

Chapter 2

Rules and Game Progression

In this chapter we will explain the precise rules of the game. The game is played with three to five players and uses the following cards:

- 40 animal cards, ten sets of four animals each. Figure 2.1 and Table 2.1 list these chards. Each card shows how much the set is worth when completed.

Figure 2.1: A picture of every animal card in the game.

Animal Name	Set value
Chicken	10
Goose	40
Cat	90
Dog	160
Sheep	250
Goat	350
Donkey	500
Pig	650
Cow	800
Horse	1000

Table 2.1: All animal cards and their values.

- 55 money cards, representing a specific value of coins. These cards are:
  - 10 times with a worth of 0 coins
  - 20 times with a worth of 10 coins
  - 10 times with a worth of 50 coins
  - 5 times with a worth of 100 coins
  - 5 times with a worth of 200 coins
  - 5 times with a worth of 500 coins

  Figure 2.2 shows a picture of these money cards.

Figure 2.2: The money cards in the game.

2.1 Start of the game

At the start of the game all animal cards are shuffled and placed face down on the table. All players receive the following money cards:

- two money cards with a value of 0
- four money cards with a value of 10
- one money card with a value of 50

These cards will be held in the hand of the player, so it is not possible to know which money cards other players possess. After this the game starts. There is no rule to determine who starts first, so the players will have to resolve that themselves.

2.2 Turn

When a player starts their turn, he or she[1] will become the *auction master* for the rest of the turn. He can choose to either start an auction or do a cattle trade.

2.2.1 Auction

If the deck is not empty, the player can choose to start an auction. He draws a card from the deck and places it face up on the table.
If the player draws a donkey card, all players receive money, depending on how many donkey cards have been drawn:

- If it is the first donkey card, everyone receives one money card worth 50 coins.
- If it is the second donkey card, everyone receives one money card worth 100 coins.
- If it is the third donkey card, everyone receives one money card worth 200 coins.
- If it is the fourth donkey card, everyone receives one money card worth 500 coins.

Starting with the player to the left, all players excluding the *auction master* will (clockwise) take turns making a bid on the animal, or passing. If a player has passed he is no longer allowed to bid on the animal during this turn.
Players are allowed to bid more money than they actually have in their hands. If a player actually wins the auction round bidding more money than he has, the auction round starts from scratch. In our implementation of the game we do not allow players to bid more money than they actually have.
After all players but one have passed their turn to bid, the final price of the animal is set to the highest bid. The *auction master* can now choose to buy the animal or let the highest bidder take it. If the *auction master* chooses to buy the animal card, he has to pay the final price in coins to the highest bidder. If he does not choose to buy the animal card, the highest bidder pays this amount to the *auction master*. If a player cannot pay with the right change, for example when the player only has a money card with a value of 50 coins but the price is 40, he will have to pay more. In this case he would have to pay with his money card worth 50 coins. The animal card goes to the player who won the auction round. He has to place it face up on the table in front of him, so that everyone can see what animal card each player has.

[1] From here on we will refer to a player as he.

2.2.2 Cattle trade

The *auction master* can also choose to start a trade with another player for a specific animal card. He can only do this if he possesses an animal card that both he and the other player possesses. The winner of the cattle trade will receive the chosen animal card from the losing player. This means that losing the cattle trade results in the *auction master* also losing his own copy of the animal card. If both players possess two of the animal cards, the trade will be for both of the animal cards. This means that the winner of the trade will get both cards from the loser, completing his set in the process. Since you cannot start a cattle trade for a specific animal card without owning at least one copy of it yourself, the set will remain his for the rest of the game.
The *auction master* starts by pointing out the animal card he is trading for to his chosen opponent. He then places one or more money cards to his choosing face down on the table in front of him. The other player then has the choice to accept the bid or to place a counter-offer (in both cases usually not knowing how much money the other player actually placed). If he accepts the trade, he takes the offered money cards and hands over the animal card(s) to the *auction master*. Other players are not able to see which money cards were offered by the *auction master*.
If he does not accept the offer, he also has to place one or more money cards to his choosing face down on the table in front of him. Both players receive the money cards the other player had placed, and the animal card(s) goes to the player who ended up offering the most money. Other players are not able to see which money cards were traded. If it was a tie, both players will have to make a new offer by placing a new set of money cards face down on the table. If this results in another tie, the *auction master* receives the animal card for free. In our implementation of the game, ties always result in the *auction master* receiving the animal card, even if it was the first tie.
After the trade or auction round the player to the left of the *auction master* starts his turn.

2.3 Game end

The game ends when all animal sets are completed. The score for each player is determined as follows: every player counts the number of sets they have, and calculates the sum of the value their sets are worth. They multiply the number of sets with that sum and add that to their score. The player with the highest score wins the game. There is no rule specified in the rule book to determine who wins in case this results in a tie, likely due to this being extremely rare.

2.4 Variations

There are a few variations to the Dutch and German version of the game:

- Instead of drawing one card in the auction round, the *auction master* has to draw multiple cards. The amount of cards drawn has to be determined at the start of the game.

- Make bidding more costly by requiring players to overbid a minimum amount of money.
- If during a trade both players possess two of the animal cards, still trade for one of them instead for the whole set.
- Add the worth of the remaining money cards to the final score each player has.

Chapter 3

Related Work

During our research we were not able to find any related work done on this game. However, we did find studies on relevant artificial intelligence techniques, like the Monte Carlo Tree Search algorithm. It is a variant of the Monte Carlo algorithm and was evaluated in a paper by Cameron Browne et al. [3]. Since it automatically visits interesting moves more often, it might be useful to implement it in this game in order to speed up the Monte Carlo based agent.

A big part of the game is that players have to deal with hidden information, on which a PhD thesis was written by Daniel Whitehouse [7]. A variant on the Monte Carlo Tree Search (MCTS) algorithm, Information Set MCTS, was proposed. It was shown to be competitive to human players in certain games and was able to deal with bluffing to certain extent. The algorithm was successfully applied to the game The Resistance, but in other games it would require an infeasible amount of computational resources to deal with bluffing.

Since the game is meant to be played against human players, an opponent modelling algorithm could be considered. For example, such an algorithm could be used in order to predict how likely opponents are to counter a cattle trade. In the paper 'Opponent modeling in poker' by Darse Billings et al. [2] they propose a solution to this for the game Poker, a popular card game with similar elements like hidden information and bluffing. They expand upon this by creating an Artificial Neural Network in order to identify more variables which are useful to predict opponents' behaviour [4]. However, such an algorithm would be difficult to apply to Kuhhandel since there is no data available on how humans play the game.

Chapter 4

Implementation

In this chapter we describe how we implemented the three different artificial players.

4.1 Deviations from the original game

Our implementation of the game differs slightly from the one as specified by the rule book. The following changes have been made in order to make the game easier to implement:

- Each game is played with four players.
- During an auction round players are not allowed to bid a value of money they cannot afford to pay if they win. However, according to the rules this is a legal move and can be used to make an opponent back out or pay a premium price for an animal card.
- During a cattle trade it is possible for both participating players to offer the same value of money cards. When this happens in our game the *auction master* wins the trade. According to the rules both players would have to make a new offer, which the *auction master* then wins if this results in another tie.

4.2 Determinization of money cards

Each player can have some knowledge over what money cards each opponent possesses. They keep track of what money cards their opponents have using the following rules:

- After an auction round the winning player has to pay the losing player[1]. All players now have knowledge of the money cards the losing player received, which they add to their list of known money cards of the losing player. They also subtract these money cards from the list of known money cards of the *Auction Master*, since he might have used those in order to pay the highest bidder.

[1] The *Auction Master* versus the highest bidder, or the other way around.

- After a cattle trade the players involved in the trade will first subtract the received cards from the list of known money cards of the other player, after which they will add the money cards they gave away to that list. All other players have no knowledge of what cards were traded and thus have to reset their list of known money cards about both players involved in the cattle trade.

By adding these rules we expect the Monte Carlo and Rule-Based agents to perform significantly better. However, no evaluations were done to confirm this.

4.3 Phases

KUHHANDEL is no ordinary turn based game where turns are easily defined like in chess, where you are only able to move a piece (or sometimes two) to another square. Based on the choice of the *auction master* at the beginning of a turn, a whole different type of turn will be started, always requiring multiple players to make decisions. Therefore, we will break the decision making down into multiple parts, as shown below. Each option is mutually exclusive to each other and a player cannot pass his turn, unless he cannot make a legal move[2]. We have the following options:

1. The current player is the *auction master* and has to choose to start either an auction round or a cattle trade.

   (a) The *auction master* chose to start an auction round. The other players take turns bidding money and a final price is set when all but one player has passed. The *auction master* now has to choose whether to buy the animal card from the highest bidder or not.

   (b) Or the *auction master* chose to start a cattle trade. He now has to choose a card from another player to trade for. He also places some money cards face down on the table. Since ties result in the *auction master* winning the trade, no further decision making is required.

2. The current player is not the *auction master*, and has to wait for the *auction master* to make a decision.

   (a) An auction round was started by the *auction master*. The players take turns bidding money. As long as the current player has not passed he has to decide whether to overbid or to pass, forfeiting the animal card.

   (b) A trade was started by the *auction master* with the current player. He has to decide whether to accept the initial offer or not.

      i. The initial offer was accepted and the round ends.

      ii. The initial offer was rejected and the current player has to make a counter offer.

Figure 4.1 shows a chart of the decision making process based on the phases specified above.

[2]This happens when the deck of animal cards is empty and the player only possesses complete animal sets.

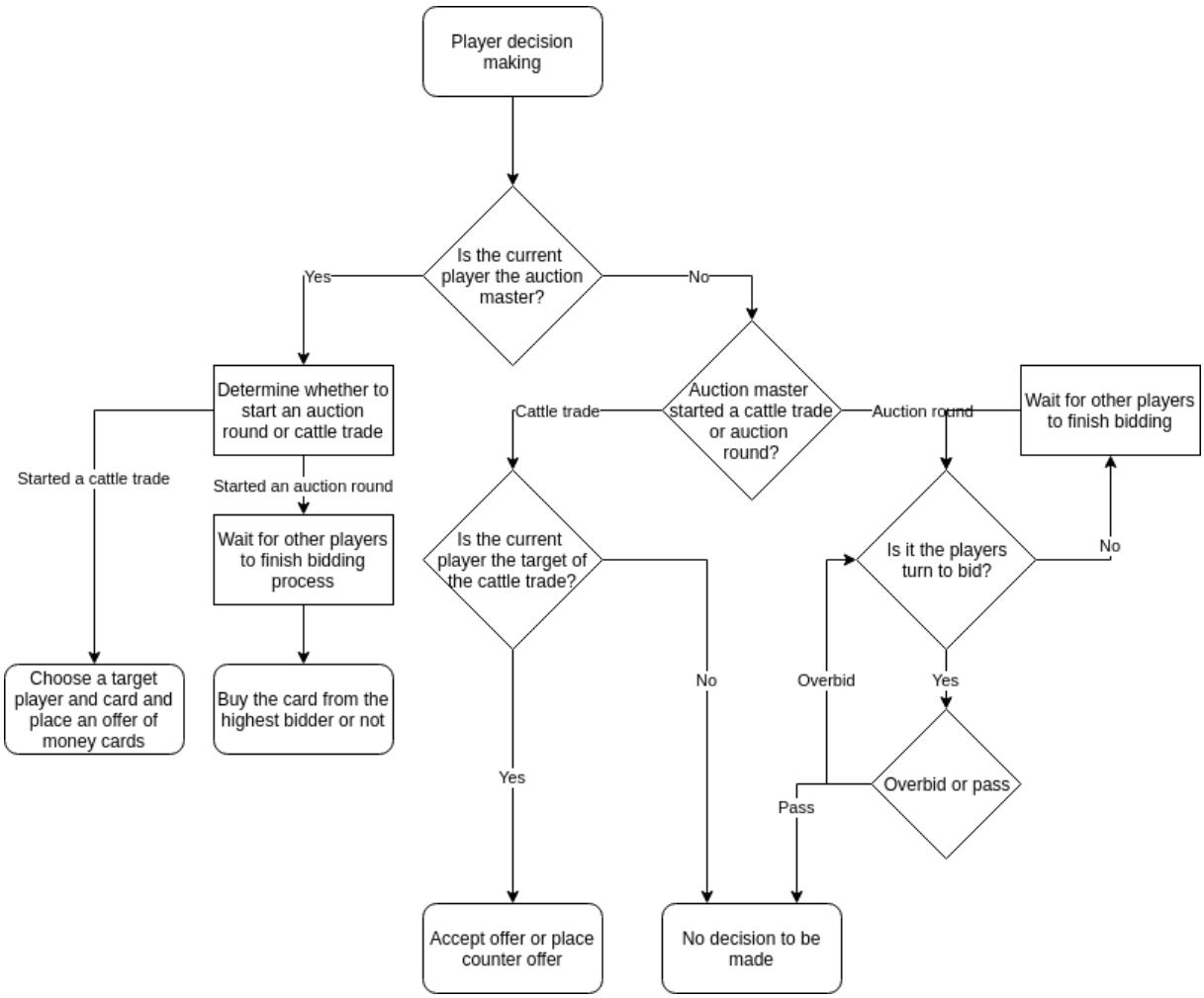

Figure 4.1: Chart of the decision making process.

4.4 Random player

The Random player makes all his choices randomly. All choices the player can make as defined in Chapter 4.3 have an equal chance to be chosen, except for the following cases:

- When it is the agents turn to bid, he will either overbid or pass. There is 30% chance that the player passes. This value was set through trial and error. A lower percentage made the agent too eager to bid on animal cards, bidding most or all of his money cards every time an auction round started. A higher percentage meant that the final price would be set very low, since the other Random agents would drop out too fast. If the player does not pass, he will overbid a multiple of 10 coins. The multiple is determined by repeatedly having a 50% chance to add another 10 coins to the bid, as long as the player can afford to do so.
- When the *auction master* offers a trade to the random player, there is a 30% chance he will accept the initial offer, since it is usually preferred to make a counter offer. This is, however, dependent on how many money cards were offered in the first place.

The Random agent will be used to compare how much better the Monte Carlo agent and the Rule-Based agent perform.

4.5 Monte Carlo player

The Monte Carlo algorithm is a technique in which a numerical result is obtained by repeatedly making random decisions [5]. For our implementation of a Monte Carlo based agent this means that for each possible move several games are played out until the end, using the Random player to make decisions for each player. The amount of games played per move can be varied in order to find the optimal balance between the speed and the performance of the agent.

At the start of each simulation we give each player, other than the Monte Carlo player, new money cards. Money cards of which the Monte Carlo agent knows who has them are not shuffled. Each player will keep the same amount of money cards they had in the first place.

The resulting score of these simulations is averaged and saved. This way we can assign an expected score for each move and keep track of the best one. Since the way you calculate the score of a game affects how the Monte Carlo player performs, we will compare two different scoring methods:

- Simply take the score as calculated by the game rules. The goal is to maximize one's own score.
- Take the score as calculated by the game rules and subtract the score of another player. Do this for each other player as well and average the results. The goal is to maximize the difference in score between the Monte Carlo player and all other players.

Since this game has different types of moves, calculating each possible combination of events from the start of the turn is too expensive to calculate. Therefore we implemented the Monte Carlo algorithm at different *levels*, similar to the different phases as defined in Chapter 4.1. In addition to this we decrease the amount of games played per move when there are more than 400 possible moves. In this case, the amount of simulations done per move is defined as: $(400 * SimulationsPerMove)/MoveSetSize$, with $SimulationsPerMove$ being the predefined number of simulations that will be done for each possible move and $MoveSetSize$ being the amount of moves the agent is able to do at the current game state.

The $MoveSetSize$ usually reaches the threshold of 400 possible moves when the Monte Carlo player has collected a vast amount of money cards, since a Cattle Trade requires the Monte Carlo player to go through each possible combination of money cards.

4.5.1 As auction master

If the Monte Carlo player is the *auction master*, he will start the turn by playing a predetermined amount of moves for either:

- Starting an auction round and never buying the card from the highest bidder.
- Starting an auction round and always buying the card from the highest bidder.
- For each possible trade[3], go through each possible combination of money cards to offer.

This will result in either starting an auction round (determining later whether to buy the card or not) or starting a Cattle Trade.

Auction round

After the other players are done bidding we end up with an animal card and a price (the highest bid). We now simulate a predefined amount of moves for either buying or not buying the card and choose the option which resulted in the highest score.

Trade round

In our implementation we left out ties to make the game easier to implement. Since we already determined the trade we wanted to do before we started the trade round, no more work is needed here.

4.5.2 Another player as auction master

If the Monte Carlo player is not the *auction master*, we still need to make decisions.

Auction round

After the animal card has been drawn each player takes turn bidding on it. When it is the turn of the Monte Carlo player to bid, he will either choose to increase the current highest bid by 10, or to pass. For each of these choices we will play a predetermined amount of games to see which one will benefit us the most.

Trade round

If the *auction master* chooses to trade with the current player, we will construct a move-set consisting of accepting the offer or counter offering every possible combination of money cards. For each of these moves a predetermined amount of games are simulated in order to choose the best option.

4.6 Rule-Based player

The Rule-Based player will determine what to do by a predefined set of rules.

[3]For each animal card that both the Monte Carlo player and any other player have at least one of.

4.6.1 Having an edge

We define two different types of edges a player can have over another player:

- An edge in the amount of money cards. This is calculated by simply subtracting the amount of money cards the Rule-Based player has with the amount of money cards the other player has. This is useful, since it makes it easier to force a player to give up either an animal card or a high value money card[4] . It also makes it more difficult for the other player to predict the sum of coins the Rule-Based player has in his hand.
- An edge in the expected sum of coins the players have. This estimation is calculated as follows: First, the average worth of all unknown money cards[5] still in play is calculated. The expected sum of money cards the other player has is then set to multiplying the average worth of all unkown money cards by the amount of unkown money cards the other player has, and adding sum of known money cards in the other players hand to that.

4.6.2 As auction master

If the Rule-Based player player is the *auction master*, he will start by determining if he has an edge in the number of money cards over another player. Ideally we would try to keep track of who possess what money cards more precisely. Especially the money cards worth 500 and 200 coins can be the deciding factor when trading. When playing against real opponents this is easier, since real players talk to each other and react based on whether a trade was favourable to them or not. In our digital version of the game however, there are no clues which might hint what money cards players receive during a cattle trade. This means that there is little knowledge over what money cards other players possess. At this point in the decision making process we do not yet take into consideration if the other player still has a high value money card while the current player does not. Since the trade in that case might still be favourable by winning these high value money cards in the trade. This will be taken into consideration when the *auction master* actually makes the cattle trade.

If the deck is not empty and there is no edge in the amount of money cards against any other player, the player will always choose to start an auction round. Otherwise he will try to find a favourable trade by going through the list of players, sorted by the edge in the amount of money cards the *auction master* has over them, and making a trade when one of the following criteria is met, in order of importance:

1. We can complete a set by winning the trade. This option will not be considered if the target of the cattle trade is expected to have more money than the *auction master*.
2. We can take the last animal card the other player has of that type, preventing them from doing a cattle trade when it is their turn to be the *auction master*. The other player is also expected to place a higher counter offer because of this.

[4]Money cards worth 200 or 500 are rare and considered high value money cards.

[5]The unknown money cards are all money cards that are in play excluding all the money cards the Rule-Based player has assigned to a specific player, including himself.

3. Purely based on the Edge. Having more money cards than the other player makes it easier to force them to give up either their animal card or a high value money card.

The downside of this approach is that the player does not take into account if making the trade leaves him vulnerable against a trade from another player or for another animal, since winning a trade almost always results in having less money than he had before. This is really difficult to predict, since it is unknown if the player he is trading with is going to accept the initial offer, which means losing the whole offer to the opponent, or if he places a counter offer, which would minimize ones losses since you would receive the money cards the opponent had offered.

Auction round

Since the Rule-Based player is the *auction master* we only have to decide whether we want to buy the card or not. Depending on how many money cards and how many of the animal card both the Rule-Based player and the highest bidder possess we make a different choice:

- The Rule-Based player already has one of the animal card and the highest bidder already possesses two. In this case the Rule-Based player only buys the card if it leaves him with an edge in the expected sum of coins over the highest bidder. This is unlikely since the Rule-Based player would have to buy the card from the highest bidder. If it doest not leave him with an edge it makes no sense to buy the animal card, since it would mean the the highest bidder just takes both of the animal cards on his turn by starting a cattle trade.

- The highest bidder already has three of the animal card and can thus complete his set. In this case we also check if it leaves us with an edge in the expected sum of money cards over the highest bidder and make our decision based on that. If the card is particularly cheap it might still be better to buy it, since we would expect the highest bidder to offer a significant amount of money cards on his turn during a cattle trade. However, the price of the animal card is expected to be high, since other players are likely to have jacked up the price during the auction round with the knowledge that there is a player that can complete a set of animal cards if he wins.

- The Rule-Based player already has three of the animal card and can thus complete his set. In this case the Rule-Based player always buys the animal card.

- The Rule-Based player already has two of the animal card and the highest bidder possesses one. We never buy the animal card in this case, since the highest bidder will have to pay the Rule-Based player for the animal card. This always leaves us with a bigger edge over the highest bidder, which we can exploit during our next turn with a cattle trade.

- In all other cases we take a look at the value of the animal card, since there is no threat of any other player completing a set. The value given to a specific animal card is hard to determine by predefined rules, since it will vary depending on how many of the animal card were already auctioned off, how

many sets certain players have completed or are about to complete, and how many donkey cards have been drawn. Currently, the value of a specific animal card is determined as follows:

- If the Rule-Based player has none of the animal cards he only buys it if the price is lower than the value of the animal, as indicated on the animal card itself.
- If the Rule-Based player has one of the animal cards he only buys it if the price is lower than 1.5 times the value of the animal, as indicated on the animal card itself.
- If the Rule-Based player has two of the animal cards he only buys it if the price is lower than 2 times the value of the animal, as indicated on the animal card itself.

Trade round

When trading with another player the Rule-Based player will try to estimate how many coins worth of money cards the opponent has in his hand. Since the Rule-Based player has to start with an offer, he is going to assume his opponent is not going to accept the initial offer and thus will come with a counter offer. Since the Rule-Based player wants to win the trade he will offer a sum of coins equal to the estimated amount the opponent has, rounded upwards in case the Rule-Based player has no fitting change for that amount. He will also add a random amount of money cards worth zero coins to make the offer seem higher.

4.6.3 Another player as auction master

First, the Rule-Based player waits for the *auction master* to start an auction or trade round.

Auction round

During the auction round the Rule-Based player will try to assign a maximum price to the animal card. This value is set according to the following rules:

- If the Rule-Based player has none of the animal card, the maximum price is set to the value of the animal, as indicated on the animal card itself.
- If the Rule-Based player has one of the animal card, the maximum price is set to 1.5 times the value of the animal, as indicated on the animal card itself.
- If the Rule-Based player has 2 of the animal card, the maximum price is set to two times the value of the animal, as indicated on the animal card itself.

We use the same logic as used when being the *auction master* of an auction round, as described in Chapter 4.6.2. Since we have to pay the money to the auction master when winning an auction, we are especially concerned what animal cards he has. If it is safe to overbid by 10, we will do this as long as we do not cross the maximum price threshold.

Trade round

First the Rule-Based player will have to decide to accept the offer or not. Since we know how many money cards the opponent has placed, we can estimate the worth of the offer in coins using the method as described in section 4.6.1. The Rule-Based player will also assume that placing a counter offer will result in winning the trade. He always rejects the initial offer unless one of the following criteria apply:

- The Rule-Based player possesses one of the animal card. We accept if we do not have an edge in the sum of coins in this trade, unless the *auction master* already has three of the animal card and can thus complete his set.
- The Rule-Based player possesses two of the animal card but the *auction master* only has one. In this case we accept if we do not have an edge in this trade. This is to prevent losing the cattle trade, and then later on losing our last copy of the animal card as well.

The counter offer is made using the same logic as if he was the auction master, as described in Chapter 4.6.2.

Chapter 5

Experiments

The compilation of our programs was done on Windows SDK version 10.0.17134.0 using the MSVC++ 14.14 compiler. The simulations were ran on an Intel i5-2500K processor. In this chapter we discuss the results of the experiments we have done.

5.1 Monte Carlo agent

The first agent we will take a look at is the Monte Carlo agent.

5.1.1 Fitness type

The Monte Carlo agent gives each possible move a score to determine what move is best to play. Earlier in Chapter 4.5 we described two different methods of determining the *fitness* of a move:

- Simply take the score as calculated by the game rules. The goal is to maximize one's own score.
- Take the score as calculated by the game rules and subtract the score of another player. Do this for each other player as well and average the results. The goal is to maximize the difference in score between the Monte Carlo player and all other players.

We ran simulated 100 games with two Random agents and two Monte Carlo agents. The agent using the first scoring technique achieved a win rate of 53,47% whilst the the agent using the second technique achieved a win rate of 45,54%. This difference is significant enough to assume that simply maximizing ones own score, instead of trying to minimize your opponents' score, performs better when playing against Random agents.

5.1.2 Monte Carlo agents versus Random agents

When matching our Monte Carlo agent against three Random agents, the Monte Carlo agent outperformed them by a vast amount, as seen in Table 5.1.

It is notable that simulating more games per move did not make the Monte Carlo agent perform better after 50 simulations per move. We expected the agent to keep getting a higher percentage due to the large amount of moves possible[1]. Likely, this is because it is not always possible to prevent another player from completing a set, since money cards are pretty scarce. However, this might still be possible if all agents tried to do this to one specific player. We found that the Monte Carlo agent usually had six complete sets while the random agent usually had zero to three sets completed.

Table 5.1: Win percentage compared to the amount of simulation done per move with one Monte Carlo agent. 100 games were played per column.

	25 simulations	50 simulations	100 simulations	200 simulations	500 simulations	1000 simulations
Monte Carlo agent	86%	92%	93%	94%	91%	94%
Random agent 1	4%	3%	5%	0%	5%	4%
Random agent 2	5%	3%	1%	1%	4%	2%
Random agent 3	5%	2%	1%	5%	0%	0%

The performance of the Random agents got worse when adding two Monte Carlo agents to the game, as seen in Table 5.2.

Table 5.2: Win percentage compared to the amount of simulation done per move with two Monte Carlo agents. 100 games were played per column.

	200 simulations	500 simulations
Monte Carlo agent 1	52%	53%
Monte Carlo agent 2	47%	46%
Random agent 2	0%	0%
Random agent 3	0,63%	0,85%

In this case, it was very rare for a Random agent to win a game. We also see an increase in the first player winning the game, implying that the first player has an advantage against other players.

5.1.3 Performance

The speed of the Monte Carlo algorithm was fairly slow. Even with the added limitation of decreasing the amount of simulations per move when the move-set becomes very large, completing a single game with just one Monte Carlo AI agent took a long time. Figure 5.1 shows the how the time taken increases when varying the amount of simulations done per move.

We think that this could be improved by changing the way the Monte Carlo player calculates the best cattle trade possible, by not even considering combinations of money cards which are obviously bad or similar to combinations that were already tried. Performance could also be improved by distributing the amount of simulations done per move more efficiently.

[1] We encountered a move-set size of over 14,000 possible moves during a trade.

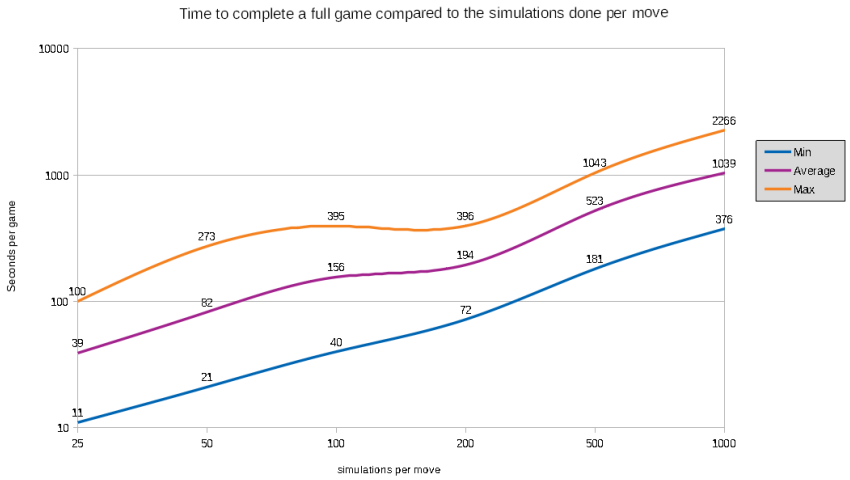

Figure 5.1: Time to complete a full game compared to the simulations done per move.

Improving performance

In order to improve the performance of the Monte Carlo player we have to consider taking a breadth-first [6, p. 81] approach of scoring our move-set instead of our current approach of evenly distributing the amount of simulated games between all possible moves.

This gives us following methods of scoring our move-sets:

1. Simulate a game for each move n times unless our move-set is larger than 400 possible moves. Distribute the amount of games simulated per move evenly. The total amount of games played is capped as if there were 400 moves in the move-set, giving us a maximum amount of moves played of $400 * n$.

2. Simulate each move one time. Take the top 100 of these moves and simulate these 10 times. Then take the best 10 of these moves and simulate them 100 times. The move with the highest score is considered the best possible move.

The first method was used in all previous testing.

We simulated 200 games using the second method of scoring our move-set. The time it took to complete a game went down the the levels of giving each move 50 simulations. However, the win-rate of the Monte Carlo agent dropped from 92% to 89%. In order to improve the win-rate we increased the amount of simulations done per move in the first round from one to three simulations per move. The increase in time was negligible, but the win-rate did not increase, lowering to 87.50%.

We think this decrease in performance is due to the agent discarding good moves by chance due to the high branching factor. This requires us to increase the minimum amount of simulations per move.

Table 5.3: Time to complete a full game using a DFS approach

Initial simulations per move	Quickest time	Average time	Slowest time
1 simulation	26 seconds	78 seconds	246 seconds
3 simulations	33 seconds	83 seconds	290 seconds

5.2 Rule-Based player

We will now discuss the results of the tests we did with the Rule-Based agent.

5.2.1 Rule-Based agents versus Random agents

The Rule-Based agent did not perform as well as the Monte Carlo agent, achieving a win rate of 81% against three Random agents. However, full games were completed in less than a second. Table 5.4 shows the results of this scenario.

Table 5.4: Win rate of various agents after 1000 simulated games.

	Win rate
Rule-Based agent	81%
Random agent	5.88%
Random agent	6.90%
Random agent	6.30%

When having two Rule-Based agents and two Random agents play, they did not quite dominate as much as the Monte Carlo agents did. In Table 5.5 the results of this scenario are shown.

Table 5.5: Win-rate of various agents after 1000 simulated games.

	Win-rate
Rule-Based agent	44.40%
Rule-Based agent	48.80%
Random agent	3.30%
Random agent	3.50%

We think that the Rule-Based player performs worse since it is not able to estimate correctly which money cards each player possesses. The Monte Carlo agent does have this ability to some extent since it shuffles all unknown money cards between every player when simulating a game.

5.2.2 Rule-Based agents versus Monte Carlo agents

To confirm our earlier suspicion that the Monte Carlo agent definitely outperforms the Rule-Based agent, we also had the Rule-Based agent play against the Monte Carlo agent in several different scenarios.

Three Rule-Based agent and one Monte Carlo agent

In this scenario it is clear to see that the Monte Carlo agent outperforms the Rule-Based agent, as shown in Table 5.6. We had the Monte Carlo agent play 50 simulations per possible move.

Table 5.6: Win-rate of various agents after 100 simulated games.

	Win-rate
Rule-Based agent	18%
Rule-Based agent	27%
Rule-Based agent	15%
Monte Carlo agent	40%

The Rule-Based player did however hold up better than the Random agent, which did not achieve a win rate of over 5% when playing against the Monte Carlo agent.

One Rule-Based and one Monte Carlo player

We were curious to see what happens if we varied the depth of the Monte Carlo player. Since the Monte Carlo agent uses the Random agents for its simulations, we had one Rule-Based agent, one Monte Carlo agent and two Random agents play against each other. The results are shown in Table 5.7.

Table 5.7: Win-rate of various agents with varying simulations done per move.

	25 simulations	50 simulations	100 simulations	200 simulations
Rule-Based AI	30%	24%	24%	24%
Monte Carlo AI	64%	71%	73%	70%
Random AI 1	5%	2%	2%	3%
Random AI 2	1%	3%	1%	3%

In this case the Monte Carlo stops improving after a depth of 50 simulations per move, just like when playing against only Random agents. It is notable that the Monte Carlo agent scores significantly higher when there are Random agents in play. Likely, this is due to the Rule-Based agent being worse in exploiting the Random agents' poor decision making.

Chapter 6

Conclusions and Future Work

We have developed three different agents for the game KUHHANDEL, a card game played by three to five players with elements of bluffing and imperfect information. We have shown that using the Monte Carlo algorithm is a viable strategy to win the game, achieving a win-rate as high as 94% against three Random agents and a win-rate of 40% against three Rule-Based players. However, these results came with the cost of taking very long turns. Both the Rule-Based agent and the Random agent were quick, but were not especially good at playing the game.

Since giving each possible move the full 50 simulations takes too long, another option of stopping exploring a move early should be implemented. For example, the Monte Carlo Tree Search algorithm could be implemented to decrease the amount of computational power wasted on bad cattle trades.

Considering the speed of the Monte Carlo agent, we are curious to see how an improved version of the Rule-Based player would perform. For example, the Rule-Based player might be able to estimate what money cards each player has more accurately. At the moment we calculate an expected sum the other players should have. But, considering how much each money card differs in value, more accurate guesses could be made based on the probabilities of owning specific money cards. This is useful for almost any aspect of the game. During an auction round the Rule-Based agent might have knowledge that the *auction master* only has low value money cards and is likely to still have a money card worth 500 coins. This means that the *auction master* is not likely to buy the animal card from the Rule-Based player, since having no fitting change would have the *auction master* lose his money card worth 500 coins. Another scenario where this is useful is during a cattle trade, where knowledge of what money cards other players have is essential when making an offer.

Bibliography

[1] Koehandel. `http://www.anderspel.nl/koehandel.html`. Accessed: 18-12-2017.

[2] Billings, D., Papp, D., Schaeffer, J., and Szafron, D. Opponent modeling in poker. *Aaai/iaai 493* (1998), 499.

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

[4] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. Improved opponent modeling in poker. In *International Conference on Artificial Intelligence, ICAIoo* (2000), pp. 1467–1473.

[5] Harrison, R. L. Introduction to monte carlo simulation. In *AIP Conference Proceedings* (2010), vol. 1204, AIP, pp. 17–21.

[6] Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[7] Whitehouse, D. *Monte Carlo Tree search for games with hidden information and uncertainty*. PhD thesis, University of York, 2014.