Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Collusion in Online Poker Pays Off

Bachelor's Thesis

Benjamin Zehnder

bezehnd@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Tobias Langner, Dr. Thomas Locher
Prof. Dr. Roger Wattenhofer

June 17, 2012

Abstract

In this thesis we describe the implementation of a colluding poker robot which plays online against human players. Our robot uses the graphical user interface of the poker platform to get the information and to interact with the poker platform. We tried to get an advantage through collusion without relying on complicated opponent simulations or a lot of domain knowledge. We let play up to four robots together against human opponents at tables of six players with play and real money.

Our results indicate that our robots are playing comparatively at play money tables but bad at real money tables. Our results further indicate that collusion provides a relevant advantage at play and real money tables, which increases with every additional colluding player. In order to win at real money tables, colluding robots need a more complicated decision making process or more domain knowledge than ours had. Another result shows that human players play more carefully with real money compared to play money.

Contents

Abstract i

1 Introduction 1

2 Related research 3

3 Materials and methods 5

3.1 No limit Texas Hold'em . . . 5
3.2 Robot design . . . 6
3.3 Interface reading and interaction . . . 7
3.4 Odds calculation . . . 9
3.5 Collusion . . . 10
3.6 Strategies . . . 11
3.6.1 Big stack strategy . . . 13
3.6.2 Hutchison strategy . . . 13
3.6.3 Minimum probability based strategy . . . 13
3.6.4 Profiling and probability based strategy . . . 14
3.6.5 Minimum probability and bluffing strategy . . . 15
3.6.6 Passive collusion . . . 15
3.6.7 Active collusion . . . 15

4 Testing 17

4.1 Play money . . . 17
4.2 Real money . . . 17
4.3 Collusion countermeasures by poker platforms . . . 18
4.3.1 Technical measures . . . 18
4.3.2 Financial measures . . . 19
4.4 Ethical considerations . . . 20

4.4.1 Play money 20
4.4.2 Real money 20

5 Results **22**
5.1 Play money 22
5.2 Real money 22
5.3 Play money compared to real money 25

6 Discussion **27**
6.1 Robot performance 27
6.1.1 Play money 27
6.1.2 Real money 27
6.2 Collusion advantage 27
6.3 The difference in human behavior between play money and real money 28
6.4 Collusion and robot detection 28
6.5 Conclusion 28

Bibliography **30**

Chapter 1

Introduction

In the past decade poker became very popular and a favorite pastime for many people. The two main reasons might be on the one hand the poker tournament broadcasts on TV, which began to show the cards of all players to the audience and on the other hand the possibility to play poker over the Internet in the comfort of your chair at home. Online poker platforms do not only allow you to play poker at home but also with much less money compared to casinos. In casinos the minimum forced bets, the so called blinds, typically start at $1 whereas at online poker platforms the blinds typically start at $0.01. This is the reason why a lot more people started to play online poker on a regular basis, which led to an immense growth in the online poker business over the past few years. In the year 2010 there were over 6 millions active online poker players worldwide and online poker platforms earned rake revenue of over US$3.6 billions [1].

The huge amount of money that can be earned in the online poker business led to a lot of different online poker platforms. They are very different in terms of their player base size but all of them claim to have robot detection and to be secure. Robots are computer programs which pretend to be humans and try to be better in poker than the average human. The security of an online poker platform is very important because the players have to trust the poker platform enough to transfer their money to the poker platform. Without a money transfer, the players are not able to play for money on the poker platform and the poker platforms can not make a profit except for advertisements.

On the other side there are several articles about robots on those poker platforms. The advantage of robots is their huge computing power which can be used to play stochastically optimal and their capability to remember histories perfectly. One player can have many robots playing at the same time without any player interaction. This possibility allows the player to make a lot of money with robots where each robot makes a little money. Many poker players do not like to play against robots because they fear to lose money against them so they prefer to play on poker platforms without robots. That is the reason why most poker platforms forbid robots.

Another problem of online poker are players which work together by exchanging information at the same poker table. This is called collusion. Since poker is a single player game collusion is considered cheating.[1] Colluding players try to cheat on fair players to get their money. Collusion is hard to detect on online poker platforms because the players can use other channels to communicate like telephones or Internet chat rooms. But for human players it may be hard because of bad communication and bad scaling in the number of players because the communication gets more complicated if there are more players.[2]

The biggest problem for poker platforms may be the combination of the two. Colluding robots that cheat on fair playing humans. The bad communication problem and the scaling problem of colluding human players do not exist if robots are used. We want to check in this bachelor thesis if the claims from the poker platforms, to detect robots and to be secure, are trustworthy. We want to do this by first building a colluding robot and then let several colluding robots play at the same time. The colluding robots will first play at play money tables and then at real money tables. This will allow us to compare human behavior between play and real money. Our goal is to show that our robot has an advantage through collusion without relying on complicated calculations and simulations or a lot of domain knowledge.

So our three main questions are: Can robots win real money on online poker platforms? Does collusion provide a relevant advantage? Do human players play differently with play money compared to real money?

[1] http://en.wikipedia.org/wiki/Cheating_in_poker

[2] http://www.pokercheatingsystem.com/articles/Article-Online-Poker-Cheating-Why-Collusion-Doesnt-Work.htm

Chapter 2

Related research

A lot of research on poker has been done in the past. Most of it focused on the two player version, the so called heads-up [2, 3]. Because our goal is to build a colluding robot we obviously cannot use the two player version. As we expect that the computational effort increases with the number of opponents at the table we decided to use the six player version and not the eight to ten player versions. Most prior research used the limited betting version of poker as this limits the number of possible actions that a player can take [2, 4, 3, 5, 6, 7]. As no limit Texas Hold'em is currently very popular we decided on that poker version.

Gilpin and Sandholm introduced a new way to compute an abstraction of the whole game tree. They showed that their approach was at least as strong as the best robot in heads-up limit Texas Hold'em at that time [2]. We cannot use this approach because the size of the game tree would be much bigger with six players and the no limit version of Texas Hold'em. So the computation would take to long to respond in the time given by the poker platform.

Billings et al. built poker robots based on opponent modeling for limit Texas Hold'em with eight to ten players. They tested their robots online against human opponents at play money tables. Their results showed that their robot played "reasonably strong" [4, 7]. We do not build a robot that does as much opponent modeling or game tree simulation but we focus on collusion and the advantage of it.

Simm built a colluding poker robot for no limit Texas Hold'em with any number of opponents. He tested his robots offline against other robots with and without collusion. The results indicated that his robots performed badly but that collusion did provide an advantage [8]. In contrast we test our robots on an online poker platform against human opponents to see how our robots perform and if collusion provides an advantage on online poker platforms.

Watson and Rubin built a poker robot for limit Texas Hold'em with up to eight opponents. Their robot was based on case-based reasoning methodology which searches for similar cases to make a decision. They tested their robot

against other robots and on online poker platforms against human opponents at both play and real money tables. The results showed that their robot performed well at play money tables and bad at real money tables [5]. We use a different approach for decision making of our robot and focus on collusion.

Risk and Szafron built poker robots for limit Texas Hold'em with two opponents. They used algorithms to find strategies which converge to an ϵ-Nash equilibrium. Their results showed that their robots were the best in limit Texas Hold'em with two opponents at that time [6]. We again cannot use this approach because the computation would take to long to respond in the time given by the poker platform because the game state is much bigger with no limit Texas Hold'em and five opponents.

Smed et al. showed that it is possible to detect collusion in most situations by analyzing the game data and modeling player behavior [9]. As we do not want that our robots are detected we do not use active collusion, which means an aggressive playing attitude, because the change in playing behavior would be much more detectable than if we only use passive collusion.

To the best of our knowledge there does not exist any work on collusion on online poker platforms. Also there is almost no work on the difference in playing behavior of human players between play money and real money on online poker platforms. These are the two topics we focus on.

Chapter 3

Materials and methods

There exist many different online poker platforms where you can play for real money. As far as we know they all do not provide an API because they do not want poker robots to play on their platforms. This is a problem we had to solve because our robot has to interact with such a poker platform else the robot will not be able to play poker on the platform. The problem of interacting consist of two subproblems.

The first subproblem is to get the information out of the poker platform. Most poker platforms provide a client which you have to install on your computer in order to play on their platform. A few also provide a flash version to play without installing any additional software. We have chosen one of the poker platforms that also provide a flash version because if you install one of their clients, then they can scan the hard drive and check for other running programs to detect a robot. They cannot do that with flash. So we have a graphical user interface from the flash version which is intended to be read by human players. In order to get the information the robot has to analyze the graphical user interface.

The second subproblem is to act on the poker platform. In order to solve this subproblem the robot has to process information about the graphical user interface, other colluding robots and the opponents and has to make a decision. Then the robot has to press the buttons and write text in the graphical user interface accordingly. In order to collude the robot also has to communicate with other robots because collusion is not possible without communication. So the robots need a way to communicate.

3.1 No limit Texas Hold'em

In this section we give a short description of our chosen poker variation no limit Texas Hold'em.

In no limit Texas Hold'em every player gets two hand cards at the start. The two players after the dealer have to make forced bets, the so called blinds. The

first has to make the small blind and the second the big blind, which is in most cases two times the small blind. There are four betting phases. In every betting phase the players can fold, check, call, bet any amount, raise any amount or go all in. The betting phase ends if there is only one player remaining or if all remaining players put the same amount of money into the pot.

After the first betting phase, the so called preflop phase, three community cards are dealt, the so called flop. The other three betting phases together are called postflop phase. After the second and the third betting phase another community card is dealt to a total of five community cards. After the fourth betting phase the winner of the poker round is determined who then gets all the money in the pot. The winner is the player with the best five-card hand from his two hand cards and the five community cards.

3.2 Robot design

Our robot uses the following three natural phases: logging into the poker platform, choosing a table and playing at the table. Logging into the poker platform is done by first opening the web page with the flash version of the platform. Then the robot detects the position of the log in window, this is done similar to button detection which is described in section 3.3, and writes the name and the password into the appropriate fields and presses on the log in button. To end this phase the robot also chooses the correct table list, real money or play money, and sets the filters for the table list. The filters are a menu provided by the platform to find a table fitting your preferences faster. We use it to only display tables with six player seats.

The robot chooses a table by first detecting the position of the list, then reading it and scrolling through it. The robot reads with a method called optical character recognition which is described in section 3.3. Optical character recognition is needed because there is no API for the poker platform and we need a way to get the information from the poker platforms interface. The text for each table is analyzed and the first table that meets our selection criteria is chosen. Our selection criteria are the following. The number of players at the table has to be at least two and the big blind at the real money tables has to be 0.02 in any of the available currencies. Then the robot checks at the table if there is really a free seat because the table lists are sometimes out of date. If there is a free seat the robot takes it else it leaves the table searches for the next table.

Now the robot is at the chosen poker table and plays. In order to play the robot has to read the table and to detect the buttons. The appearance of one of the play buttons indicates that it is our turn to play. The current strategy then decides on the action to take by considering the available table information, ex-

changing information with the collusion server, which is described in section 3.5, and calculating the probability to win. We have designed several strategies which are described in section 3.6. The action decision from the strategy is then preformed by clicking on the respective button.

3.3 Interface reading and interaction

In order to get the information from the graphical user interface of the poker platform we do optical character recognition and button detection. As we want to do this by comparing the current screen shot to images of the characters of the graphical user interface, we need sample images of all the characters. We took all those sample images and saved them in a folder along with a description file which has a list of all the characters that are in the folder. The description file also indicates which algorithm should be used for the character set in the folder. We do this because depending on the operating system or the poker platform you may want to change the algorithm accordingly to the problem without changing the source code.

The algorithms to get the information out of the graphical user interface first take a screen shot[1] and analyze it. We use different algorithms for different optical character recognition problems depending on how the characters are graphical displayed.

On the poker platform we used to do the testing the buttons looked always absolute identical. So the first algorithm, algorithm 1, checks for a perfect matching. The algorithm does this by checking if the screen shot has a rectangle of the size of the button in which every single pixel has exactly the same color as the corresponding pixel in the sample image of the button.

If the poker platform and the operating system use anti aliasing and different displaying modes, including shadows, transparency and different backgrounds, you cannot just check for a perfect matching of the characters. Our solution to this problem is that we allow a certain difference per pixel. This difference can be indicated in the description file. The only change to algorithm 1 is on lines 8 to 10 which we replace with

if $pixelDifference(screen.getRGB(x+i, y+j), button.getRGB(i, j)) > maxDiff$ **then**
  $isCorrect \leftarrow$ **false**
end if

where *pixelDifference* is shown in algorithm 2 and *maxDiff* is the difference indicated in the description file.

If only the background is different then all character pixels have the same color f. The idea is to make all the images black and white, where all pixels

Algorithm 1 Check for a perfect matching of *button* in *screen*

```
1: for x = 0 to screen.width − button.width do
2:   for y = 0 to screen.height − button.height do
3:
4:     isCorrect ← true
5:     for i = 0 to button.width do
6:       for j = 0 to button.height do
7:
8:         if screen.getRGB(x + i, y + j) ≠ button.getRGB(i, j) then
9:           isCorrect ← false
10:        end if
11:
12:      end for
13:    end for
14:
15:    if isCorrect then
16:      return true
17:    end if
18:
19:  end for
20: end for
21:
22: return false
```

Algorithm 2 Calculating the pixel difference of pixel a and b

1: $sum \leftarrow |a.getR - b.getR|$
2: $sum \leftarrow sum + |a.getG - b.getG|$
3: $sum \leftarrow sum + |a.getB - b.getB|$
4: **return** *sum*

with the color f are white and all others are black, and then compare them. In this case we use a third algorithm where the only change to algorithm 1 is again on lines 8 to 10 which we replace with

if not $(screen.getRGB(x+i, y+j) = f) = (button.getRGB(i,j) = f)$ **then**
  $isCorrect \leftarrow$ **false**
end if

where f is the color of the characters.

For input we generate native system input events.[1] With the algorithms described above we can get the position by returning x and y instead of *true*. With the information of the position we can generate a mouse move event to that position and then a mouse click event to press the button or select a field at that position. To write into a field we again generate input events. This allows us to generate all input events a human user could generate as well. As an anti robot detection measurement we added random time delays before all input events to let the robot look more like a human.

3.4 Odds calculation

The odds calculation is a very important component. We implemented two different approaches. The first one calculates the exact odds to win against one opponent with random hand cards. This is done by counting all wins and losses over all possible combinations and then dividing wins by total count to get the probability to win. As this computation takes a lot of time for more than one opponent, we need another approach for more than one opponent. Also it takes too long in the preflop phase because the additional combinations with the three flop cards add to much complexity. As there are only 169 strategically different situations in preflop due to the indifference of the colors of the two cards we precalculated those and saved the results. With multiple opponents this is not possible, as there would be too much data to save.

We implemented the second approach to calculate the odds against multiple opponents. This is done similarly to the first approach but taking a random subset of all possibilities to be within the time limit. By the law of the large numbers this should give us an approximated value close to the exact one.

[1] In Java this can be done easy by using the java.awt.Robot class.

Both approaches need some way of determining if our or the opponents cards are better. We implemented a function that translates a set of cards to a value, which indicates how strong the cards are. These values are directly comparable. In both approaches it is also possible to exclude some cards from the calculation. This is used to adjust the odds calculation when the robots are colluding and know that the other robots hold certain cards.

As most opponents do not play with all hand cards and both odds calculation approaches assume random hand cards we also included two additional hand card condition profiles. The first profile takes only card combinations into account where at least one opponent has hand cards with at least one of the following properties: same color, connected, pair or one card above or equal to ten. The second profile takes only card combinations into account where at least one opponent has hand cards which are played in preflop in any situation by the big stack strategy described in section 3.6.1.

These profiles account for the fact that opponents will not call in preflop with bad hand cards. As each profile gives a probability including the random one the robot has three different odds in total. How these are used depends on the strategy. In general we used the minimum of these three odds.

3.5 Collusion

In order to collude the robots need some way of communicating with each other as you cannot collude without information exchange. We decided on a client/server design because of the simplicity. We also decided that the server will not decide on the actions the robots will take because this would use a lot of computational resources and if there are many robots running at different poker tables then this could lead to a too slow response of the robots. For simplicity we decided that the server does not ask the robots for information but instead the robots inform the server about updates of their state. This way the server can answer requests faster because the server has all current information of the robots available.

To select a poker table the server keeps a list of all robot names and corresponding table names. If a robot wants to select a table it first searches for one that meets its criteria and then proposes the table name to the server. As the server cannot interact with the poker platform, the server needs the table names from the robots. The server then checks the list in order to place multiple robots at the same table and responds with the table name the robot should select. The robot will then search for this table and join it.

To collude while playing, the robots update their hand cards on the server as soon as they know them. In order to be synchronized the robots also send the actual dealer position. They then request all hand cards of the other robots at the same table from the server and with that information they calculate their

odds. After the calculation they send their odds with dealer position to the server. The robots then request the odds of the other robots at the same table to have them available for the strategies.

3.6 Strategies

The strategy is the deciding component of the robot as it decides on the actions the robot makes while playing poker. A selection of the following actions from poker is available in each decision the robots strategy has to make: fold, check, call, bet, raise and all in. For the decision making the robots strategy has all possible information of the table, the odds and the collusion available. We designed several strategies to find out what works best. All of them use the collusion information to calculate the odds. All of our strategies are described in this section.

The robots choose the strategy to play randomly at the start and rechoose the strategy randomly every 30 minutes to one hour. We do this because opponents should not be able to predict the robots strategy after 50 to 100 hands played. As an anti-robot/anti-collusion detection measurement the robots do not choose the strategies uniformly at random but with a certain profile of probabilities which is deterministically calculated from the login name of the robot. So every robot has a different profile. We do this because the robots should look differently in the overall statistic else the poker platform employees can easily conclude that all of those players with roughly the same statistics have to be the same robot.

We divided the strategies into two parts. One that makes the preflop decisions and one that makes the postflop decisions. We did this because the situation with and without table cards is really different but the situation with three, four or five table cards is similar, especially in our calculations for the decision making.

In all of our strategies we used the idea of calculating the ratio ρ of the expected win amount to the expected loss amount. The idea is that if we always have a ratio greater than 1, in the cases where we do not fold, then we will win on average.

$$\rho = \frac{winAmount * odds}{callAmount * (1 - odds)} \tag{3.1}$$

where *winAmount* is the amount the robot can win if it calls. *callAmount* is the amount the robot has to pay if it calls. *odds* is the minimum of the three calculated winning probabilities described in section 3.4 because we assume the worst case.

The robot now uses $\rho > min$ to decide, if it should call, where min is a

predefined value. If $min > 1$ then we expect to win overall but only under the assumptions that all opponents play with one of the three predefined hand card condition profiles described in section 3.4. Also if they have hand cards accordingly to their profile they must not fold. So in general the assumptions are not correct. Our empirical values for *min* are listed in table 3.1. We tried several values and those in the table worked best at the play money tables.

We completed the decision making equation with a linear function

$$risk(callAmount) = \frac{callAmount}{bigBlind} * \frac{highRiskFactor - 1}{100} + 1 \tag{3.2}$$

where *bigBlind* is the big blind of the current poker table and *highRiskFactor* is a constant greater than 1. We added the *risk* function because ρ would not change if all values on the table would be multiplied with a constant factor a. This can be seen in formula 3.1 where both amounts would be multiplied with the factor a. So ρ does not include information about the relative amount to the big blind the opponent players bet. But human players tend to bet/raise more if they have better cards. The idea therefore is that the higher the opponents bet, the higher the risk that they have good cards.

Our final decision making equation is therefore

$$\rho > risk(callAmount) * min. \tag{3.3}$$

For betting/raising we have to adjust the formula 3.1 to include the additional amount the other players have to pay in the *winAmount*. For this we always assume that all others will call after our betting/raising. Of course we also have to increase the *callAmount* with the amount we bet/raise in the formula 3.1 and in the decision making equation.

In preflop we generally raise two or four times the big blind. In postflop we generally raise half the amount of the old pot or the amount of the old pot where the old pot is all the money on the table from the last betting round. This is shown in algorithm 3 for preflop and in algorithm 4 for postflop. Our empirical values for *min* and *highRiskFactor* are listed in table 3.1. We tried several values out and those in the table worked best at the play money tables.

	Preflop	Postflop
minCall	1.2	1.5
minHalfRaise	1.8	2
minRaise	2.4	2.5
highRiskFactor	2	4

Table 3.1: The empirical values which we used as constants after trying several values out at play money tables.

Algorithm 3 Preflop decision making

1: $raiseAmount \leftarrow$ amount we have to pay to raise four times the big blind
2: $raiseHalfAmount \leftarrow$ amount we have to pay to raise two times the big blind

3: $callAmount \leftarrow$ amount we have to pay to call
4: **if** $\rho > risk(raiseAmount) * minRaise$ **then**
5: **return** bet/raise the amount of the old pot
6: **end if**
7: **if** $\rho > risk(raiseHalfAmount) * minHalfRaise$ **then**
8: **return** bet/raise the amount of half the old pot
9: **end if**
10: **if** $\rho > risk(callAmount) * minCall$ **then**
11: **return** call
12: **end if**
13: **return** check/fold

3.6.1 Big stack strategy

This strategy works accordingly to Pokerstrategy.com[1] in the preflop phase of the game. It is basically a table where is listed what you should do with certain hand cards and in a certain positions depending on what the other players did. We did a few adjustments as the actions were weird in some situations. Pokerstrategy.com does not describe very well what should be done in certain situations after the flop. So we used our algorithm 4 for postflop decision making.

3.6.2 Hutchison strategy

This strategy works accordingly to the Hutchison system[2] in the preflop phase of the game. The system works by building a sum representing the strength of the hand cards. The decision making is then based on this sum. For postflop decision making we again used our algorithm 4.

3.6.3 Minimum probability based strategy

For this strategy we used algorithm 3 for preflop and algorithm 4 for postflop decision making.

[1] http://www.pokerstrategy.com/home/

[2] http://www.poker-institut.org/strategien/hutchison-punkte-system-holdem/
or http://www.pokerstrategy.org.uk/articles/preflop-hand-selection-the-hutchison-system

Algorithm 4 Postflop decision making

1: $raiseAmount \leftarrow$ amount we have to pay to raise the amount of the old pot
2: $raiseHalfAmount \leftarrow$ amount we have to pay to raise the amount of half the old pot
3: $callAmount \leftarrow$ amount we have to pay to call
4: **if** $\rho > risk(raiseAmount) * minRaise$ **then**
5: **return** bet/raise the amount of the old pot
6: **end if**
7: **if** $\rho > risk(raiseHalfAmount) * minHalfRaise$ **then**
8: **return** bet/raise the amount of half the old pot
9: **end if**
10: **if** $\rho > risk(callAmount) * minCall$ **then**
11: **return** call
12: **end if**
13: **return** check/fold

3.6.4 Profiling and probability based strategy

For this strategy we implemented a database for opponent players. This database saves the ratio of starting hands a specific opponent plays and how much he bets. It is hard to put these two values into one, as an opponent that plays rarely but bets a lot and an opponent that plays often but bets not that much will look kind of similar, even though they play totally differently. So we used two values to represent a player. To take strategy changes into account, we also make a short history which is just the recent part of the long history of a certain opponent. We compare these two histories about every 30 poker rounds and if they are not approximately the same then we delete the old part of the long history of this opponent. We do this because we do not want to have an average of several strategies when we aggregate the history and use it for decision making.

This strategy is based on the algorithm 3 and 4. Both algorithms are adjusted to this strategy by multiplying the right side of every if condition with the return value from algorithm 5. If there are several opponents then the maximum of the return values from algorithm 5 is taken.

The problem with the profiling is that it is hard to find a way to put it into the strategy and not to make it perform worse. Another problem is that you do not have the information at the beginning and you have to play a lot against the same opponent to get a large enough data set which is representing the player correctly.

Algorithm 5 Calculating the risk of the opponent

1: $opponentRisk \leftarrow$ ratio of played hands in preflop by opponent
2: $opponentRisk \leftarrow \max(0.5, (1 - opponentRisk) * 2)$
3: **if** preflop phase **then**
4: $average \leftarrow$ average bet in preflop by opponent
5: $bet \leftarrow$ bet the opponent made this poker round
6: $bigBlind \leftarrow$ big blind of this poker table
7: $relativeBet \leftarrow \max(1, bet/bigBlind)/average$
8: $relativeBet \leftarrow \max(0.5, \min(2, relativeBet))$
9: $opponentRisk \leftarrow opponentRisk * relativeBet$
10: **end if**
11: **return** $opponentRisk$

3.6.5 Minimum probability and bluffing strategy

For this strategy we used algorithm 3 and 4 as a basis and added semi bluffing to it. Semi bluffing means for us that we only bluff if the condition $odds * numberOfPlayers > 1$ holds where *odds* is our winning probability described in section 3.4 and *numberOfPlayers* is the number of poker players at the table that still hold their hand cards including ourselves. In preflop we semi bluff with a probability of 20% by reraising but only if an opponent raised. In postflop we semi bluff with a probability of 15% by betting the amount of the old pot independent of what the opponent did.

3.6.6 Passive collusion

The idea for this strategy is to use one of the other strategies but preventing two robots from betting or calling in the same round in any postflop phase. For this the robot with the highest odds that comes first sets a flag on the collusion server to be sure that another robot with the same odds will not bet or call as well. The robot only sets the flag if he does bet, raise or call. All other robots will just check or fold. The collusion information is already used by the other strategies if collusion mode is enabled. So this strategy does not have to change odds calculation of the other strategies.

3.6.7 Active collusion

The idea for this strategy is to actively use the fact that multiple robots are at the same table. One example for this would be two robots that always reraise each other in small steps if one of them is almost sure to win. This way the opponents have to decide in every step if they want to call the small amount to stay in the game or fold. If the opponents call they probably will call all

future raises as the pot was increased and the amount to call remains the same. The problem is that it is quite obvious that the robots are colluding and the opponents will report the collusion of the robots quite fast to the support of the poker platform. The support would then see the obvious collusion behavior and close our accounts. That is why we did not implement it.

In general we think it is very hard to make any active collusion strategy which is not easy detectable in the long term.

Chapter 4

Testing

In order to test our robots we let them run on Poker770.[1] We always used tables with six players and the game mode was *no limit Texas Hold'em*. We let them run alone and in collusion mode where up to four players played together. Overall we let the robots play a few ten thousand rounds of poker.

Every robot runs in its own virtual machine and uses a virtual private network connection to the servers of our university to have its own unique IP address. The robots need an own IP address because the poker platform prevents players with the same IP address from sitting at the same table.

4.1 Play money

We let the robots play at the play money tables with a big blind of 10 and a buy in of 1000. You always can reset your play money to 1000. This obviously leads to careless play of some opponent players even though it is not as bad as you might think.

4.2 Real money

We decided to play at the real money tables because of the careless play of some opponent players at the play money tables. Because of ethical considerations we played only at tables with a big blind of 0.02 and a buy in of 1.00 of any of the available currencies which were United States Dollar, British Pound and Euro.

The real money to play with we got from the poker platform itself by creating 22 faked accounts and claiming the welcome bonus of $7.70. We created the faked accounts by faking names and birth dates and taking any real addresses from Google Maps[2] in Switzerland and for the phone numbers we faked mobile

[1] http://www.poker770.com
[2] http://maps.google.com/maps

numbers, as these are not in a phone book normally. We then created a Yahoo[3] mail account for every single faked account and then registered the account at Poker770[1] with that e-mail address. When you register an account normally a customer support chat window pops up, where they want to convince you to pay in some money. If we chatted with them or at least said "no thanks" they did not make a real ID check for the welcome bonus in most cases. If they did a real ID check on an account we did not use the account anymore.

The problem with paying in some money is that the money has to come from a bank account. If you now want to collude, it probably is not advisable to pay in the money from the same bank account for the colluding robots. But there is also another problem. Every normal bank account has some owner information available, so you cannot use it for a faked poker platform account but then you need several poker platform accounts from real persons to do collusion. The question remains if the poker platform providers check if the bank account owner information corresponds to the poker platform account or not but we think they do.

We tried to use credit cards to transfer real money to the accounts but the countermeasures from the poker platform prevented us from doing so. This is described in subsection 4.3.2.

4.3 Collusion countermeasures by poker platforms

Here we present the collusion countermeasures done by the online poker platform Poker770[1] and probably the whole iPoker network[4] which we encountered during our testing. The poker platform did not detect any of our robots during the whole testing. Neither did they detect the collusion between the robots to the best of our knowledge.

4.3.1 Technical measures

The poker platform does not allow more than one player with the same IP address per poker table. This measure prevents members of the same household to play at the same table if they share an IP address. This is possible because of the current IP infrastructure of most households as they share one IP address within the local area network. With the change from IPv4 to IPv6 this measure may become useless as there is no need to share an IP address anymore. We worked around this measure by connecting to the virtual private network of our university which provided us with an unique IP address for every robot. We argue that this measure does not prevent collusion because the people that can

[3] http://www.yahoo.com/
[4] http://www.ipoker.com/html/

program a poker robot probably know how to get multiple IP addresses. But it is not easy to circumvent the IP address measure which prevents typical members of the same household from colluding with each other.

The flash version of the poker platform saves all the players that log in in the browser cache. The poker platform then uses the browser cache to prevent players that are in the same browser cache to sit at the same poker table. After we deleted the browser cache with two robots in it these two robots could play again together at the same table. This shows that the poker platform does not save that information on its servers. We think that they should save the information of the two players, that played at the same computer, server side, as soon as they detect two players in the same browser cache. This measure does not prevent collusion if the colluding players pay attention where they log in. We argue that this measure is not very useful because you can delete the browser cache and then collude. But if the information would be saved on the server then this measure would be at least a little bit better because the two players can never again play together. We have no clue why the information is not saved on the server as it could be very easily done.

4.3.2 Financial measures

The poker platform prevents the use of the same credit card for two different accounts. But it does not prevent the use of multiple credit cards from the same person for different accounts. Also there are other payment methods for example WebMoney[5] which is an anonymous payment system. We argue that this limitation of the system is not only useless but also inconvenient for households with only one credit card but multiple online poker players. It is useless because the poker platform provides itself a way around the limitation with the possibility to pay with WebMoney.[5]

The poker platform providers check if the players have registered with their real names and addresses but not at the time of the registration. We have registered 22 fake accounts at Poker770[1] and claimed the welcome bonus of $7.70. They asked 8 of the accounts for official documents of their identity before they would have provided the bonus, since we obviously could not come up with such documents, we abandoned these 8 accounts. So we had 14 accounts with $7.70 to test. We transferred money with two credit cards to two of those 14 accounts. As the names of the credit card owner and the poker account owner were not the same, the support wanted an explanation. We told them that we do not have a credit card and that this credit card belongs to a friend. After providing our fake data again we could transfer the money to the poker accounts. The day after they temporarily closed our two fake accounts until we would provide official documents of our identity. We obviously could not do that because the accounts

[5] http://en.wikipedia.org/wiki/WebMoney

were created with faked data. This indicates that the poker platform providers of Poker770[1] do always check for the real identity of their poker players if the players want to pay money in or out. We argue that this measure does work quite well but there is a way around it. You could ask some people to provide you with all the information and documents you need and then there should be no problem to do it. The problem with this approach is that you involve other people into the cheating. We did not want to involve other people so we did not do it. The question remains why the poker platform providers do not check the identity of all players who claim the welcome bonus.

4.4 Ethical considerations

We want to test our robot on a real poker platform against opponents which are normally playing there. The effort to ask them all if they want to participate in the testing of our robot would be too big. If they turned the chat off then we would have no way of contacting them because they are playing under a pseudonym. Also they might change their strategy because they know that they are playing against robots what would influence our results. Because we are doing an experiment with humans involved we have to make ethical considerations especially as they do not know that they are involved in an experiment. One might think that there is no problem because it is just a game and we only made a robot that plays instead of us but we think that it matters. We distinguish between play money and real money because we think that the situation is not really comparable.

4.4.1 Play money

The play money games should be no problem as there is nothing real to lose or win. Also a perfect robot will lose from time to time due to the chance component of poker and our robot is not perfect.

4.4.2 Real money

For real money games the situation is totally different compared to the play money games. It would be not that bad if we only used one robot per table and would not collude. Because then the robot has the same possibilities as a human player. But we try to win with cheating against human players who know nothing about our experiment. That is why we decided to play only at the lowest available big blind level which is 0.02 of any of the available currencies which are United States Dollar, British Pound and Euro. Also we decided to stop the testing in case we would win a significant amount of money. We can not give the money back to the opponent players because they are playing under

a pseudonym. So the worst that could happen would be that some players lose a few dollars, pounds or euros. As the game is chance based and some players lose a few dollars, pounds or euros against other players anyway we think this is justifiable.

The poker platform itself may lose a few hundred dollars in case the robots lose because it is their money we are playing with and if we lose it to other players they can withdraw it. We do not think that has a great impact on the company as they have a lot of players and earn a lot of rake from the players. While the robots play the company earns some of the money back as they earn the rake independent of who the winner is. In case that the robots win the company will make money as we do not intend to withdraw the money from the poker accounts. So this experiment will have a justifiable impact on the company.

Chapter 5

Results

In this chapter we present the results we obtained from the testing.

5.1 Play money

In figure 5.1 are the results of our robots at play money tables. The graph shows *win* − *loss* in play money over the poker rounds. The robots played with a buy in of 1000 and a big blind of 10. One poker round equals one robot playing in one poker round. This means that if x robots played at the same table for one round then the graph shows x rounds, one for every robot. The robots played with and without collusion with up to three other robots per table. The graph shows an increase in play money of about 80′000 in about 6′000 rounds. This is an increase of about 13 in play money per round per robot what is more than one big blind per round per robot.

Table 5.1 shows the results per strategy of the graph in figure 5.1. All strategies are winning at play money tables. The best of our strategies is the collusion minimum probability strategy with an average of about 25 in play money per round per robot what is about two and half times the big blind per round per robot.

In table 5.2 is a comparison between the different numbers of colluding robots at one table. The data is the same as in figure 5.1. The results show an increase on average per robot if more robots play at one table.

5.2 Real money

In figure 5.2 are the results of our robots at real money tables. The graph shows *win* − *loss* in real money over the poker rounds. The robots played with a buy in of 1.00 and a big blind of 0.02 in any of the available currencies which are United States Dollar, British Pound and Euro. One poker round equals one

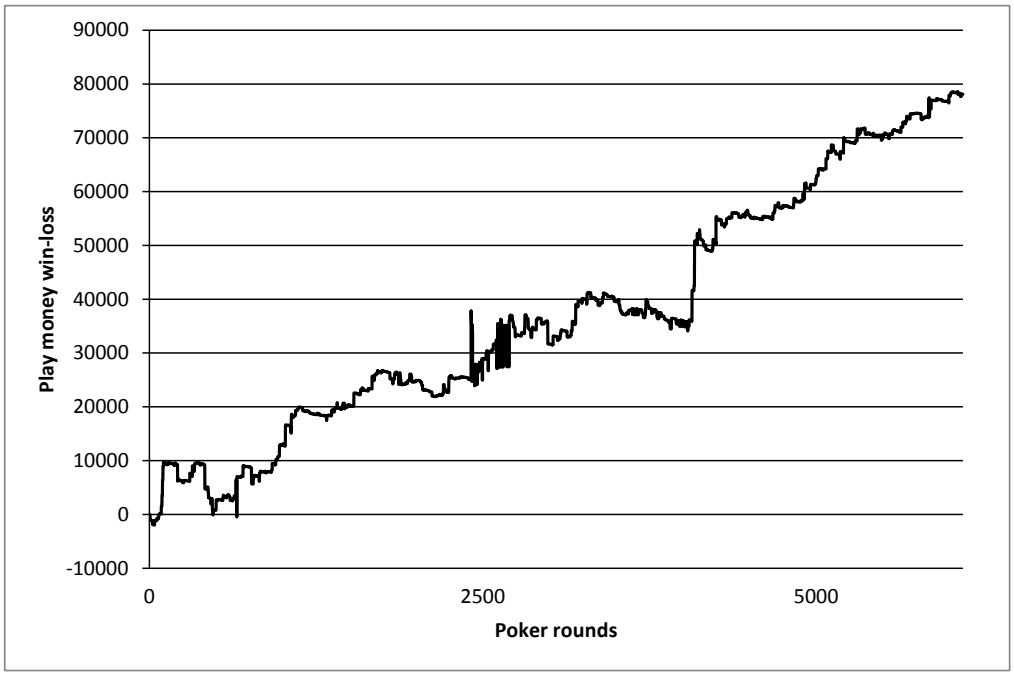

Figure 5.1: Play money $win - loss$, big blind = 10, buy in = 1000, 6 players per table, with and without collusion with up to four robots per table

Strategy	Average $win - loss$	Count	Total
Big stack	13.52	983	13294
Hutchison	5.51	308	1698
Minimum probability	18.70	717	13407
Profiling and probability	5.12	127	650
Collusion big stack	3.51	1419	4975
Collusion Hutchison	15.90	191	3038
Collusion minimum probability	25.04	1345	33684
Collusion profiling and probability	13.08	700	9155

Table 5.1: Play money $win - loss$ per strategy, big blind = 10, buy in = 1000, 6 players per table, with and without collusion with up to four robots per table

Number of robots	Average $win - loss$	Count	Total
1	9.66	4178	40377
2	19.35	1201	23240
3	24.18	690	16687

Table 5.2: Play money decision making compared between the number of colluding robots at one table, big blind = 10, buy in = 1000, 6 players per table

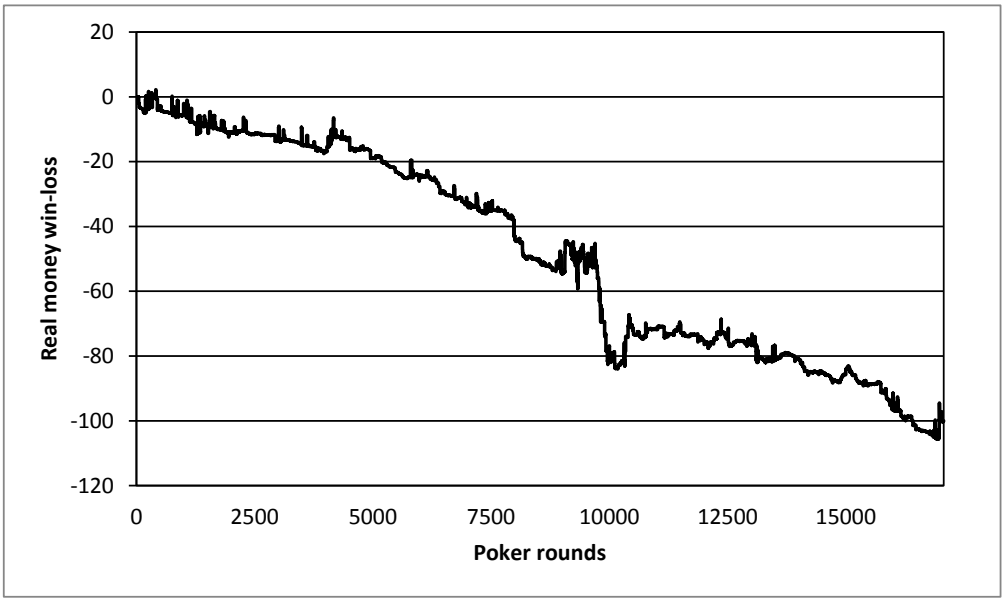

Figure 5.2: Real money $win - loss$, big blind = 0.02, buy in = 1.00, 6 players per table, with and without collusion with up to four robots per table

robot playing in one poker round. This means that if x robots played at the same table for one round then the graph shows x rounds, one for every robot. The graph shows a loss of about 100 in about 17′000 rounds. This is a loss of about 0.006 per round per robot what is about a quarter of a big blind per round per robot.

Table 5.3 shows the results per strategy of the graph in figure 5.2. All strategies are losing at real money tables. The best of our strategies is the collusion minimum probability strategy with an average of about −0.005 per round per robot.

In table 5.4 is a comparison between the different number of robots at one

Strategy	Average $win - loss$	Count	Total
Collusion big stack	−0.00523	4604	−24.08
Collusion Hutchison	−0.00648	1376	−8.92
Collusion minimum probability	−0.00465	2244	−10.44
Collusion profiling and probability	−0.01310	145	−1.90
Collusion min. prob. and bluffing	−0.00545	8468	−46.19

Table 5.3: Real money $win - loss$ per strategy, big blind = 0.02, buy in = 1.00, 6 players per table, with and without collusion with up to four robots per table

Number of robots	Average $win - loss$	Count	Total
1	−0.0069	2788	−19.15
2	−0.0092	6124	−56.38
3	−0.0043	5172	−22.03
4	−0.0010	2950	−3.08

Table 5.4: Real money decision making compared between the number of colluding robots at one table, big blind = 0.02, buy in = 1.00, 6 players per table

Take only results into account with an absolute value change greater...	Play money	Real money
10 times the big blind	0.0887	0.0485
20 times the big blind	0.0693	0.0354
30 times the big blind	0.0588	0.0296
40 times the big blind	0.0523	0.0269
50 times the big blind	0.0461	0.0214
Total rounds	6102	17069

Table 5.5: Number of decisions with an absolute value change of money greater than 10 to 50 times the big blind

table. The results show an decrease on average from 1 to 2 robots but an increase on average from 1 to 3 and 3 to 4 robots. The decrease from 1 to 2 robots is a result of the parameter tuning for real money tables which we did while 2 robots were playing.

5.3 Play money compared to real money

Table 5.5 compares the ratio of big decisions to all decisions made by the robots between play money and real money. Big decisions change the robots amount of money by an absolute value greater than 10 to 50 times the big blind. The results are relevant because the opponent players had to call the bets of our robots or our robots had to call the bets of the opponent players to count since the robots lost or won at least 10 to 50 times the big blind. The results show that big decisions appear almost twice as much in play money games compared to real money games.

Table 5.6 compares play money and real money in respect to the reached betting phase. The results show that the last betting round is reached almost in three out of four games at play money tables while at real money tables the last betting round is reached in only about one out of five games.

	Play money	Real money
Total rounds	6102	17069
Ratio of rounds with 0 table cards	0.076	0.473
Ratio of rounds with 3 table cards	0.110	0.237
Ratio of rounds with 4 table cards	0.091	0.086
Ratio of rounds with 5 table cards	0.723	0.204

Table 5.6: This table compares play money and real money in respect to the reached betting phase.

Chapter 6

Discussion

In this chapter we discuss our results and make a conclusion.

6.1 Robot performance

6.1.1 Play money

As figure 5.1 shows our robot performs very well at play money tables. The tables 5.1 and 5.2 show that the robot performs also very well without collusion as the robot does not collude if he is not using any of the collusion strategies. This indicates that our robot is comparative at the play money level.

6.1.2 Real money

Figure 5.2 shows that our robot performs badly at real money tables. We tried to adjust the parameters to the real money tables but no parameter configuration we tried worked. We think that the difference to play money tables comes from the difference in the playing behavior by the human players. The simple approach to calculate the ratio ρ of the expected win amount to the expected loss amount that is described in section 3.6 does not seem to be enough to win at the lowest level of real money tables.

6.2 Collusion advantage

Tables 5.2 and 5.4 indicate that collusion provides a big advantage. Even though we only used passive collusion the results with play money show a huge increase in average win. At real money tables our robots also profit from collusion, except with two robots, even though they are still losing. The result of 2 robots is worse than the result of 1 robot because we tried to tune the parameters for the real money tables while 2 robots were playing. The result where 4 robots

are colluding at real money tables with an average of -0.001 is five times better than an always fold robot would be since the always fold robot would pay one big blind and one small blind per six rounds what equals $-0.03/6 = -0.005$.

We asked ourselves at the beginning if collusion provides a relevant advantage. We argue that the results indicate that the answer is yes as the tables 5.2 and 5.4 show a better decision making with the additional knowledge from collusion.

6.3 The difference in human behavior between play money and real money

Table 5.5 indicates that human players are much more careful when playing at real money tables because they can lose real money. The table shows that human players are much less willing to risk 10 to 50 times the big blind when playing at real money tables compared to play money tables. We think that the reduction to almost half as much played big decisions shifts the winning from the robots to the human players.

Table 5.6 shows that human players play much less hands while playing for real money even though we played at the lowest level. Almost half of the poker rounds at real money tables already end in the preflop phase while at play money tables only 7.6% end in the preflop phase. This is also due to the more careful play by the human players.

6.4 Collusion and robot detection

The poker platform did not detect any colluding robot or the collusion. We do not know if they check at all or only real money tables. Perhaps they only check if a player wants to withdraw money. As our robots lost real money we cannot say much about the detection because if someone loses while cheating it does not really matter in our opinion and play money does not really matter as well.

6.5 Conclusion

We built a colluding poker robot for no limit Texas Hold'em with six players. We used several different approaches to read out the interface and we designed a few strategies which use an equation with the ratio of the expected win amount to the expected loss amount for the decision making. This approach uses very little domain knowledge and is very fast except for the odds calculation. Our goal was to get an advantage through collusion and not a complicated decision making

process. In order to test our robot we let it run on the poker platform Poker770[1] at both play money and real money tables. The poker platform providers did not detect any of the colluding robots or the collusion even though they claim to detect both.

Our results indicate that our robot is competitive at play money tables but not at real money tables. This shows that collusion alone is not enough to win at real money tables. We think that with more complicated calculations or a lot of domain knowledge it should be possible to build a winning robot for real money tables. Our results also indicate that collusion gives a relevant advantage in decision making, which increases with every additional robot. Another result of our research is that human players play very differently if they play with real money compared to play money. The huge difference in playing behavior shows that it is not sufficient to test robots against human opponents at play money tables.

Future research has to be done on the psychological difference for humans to play with real money compared to play money and on the question if robots can earn money at online platforms and if so if they are detected.

[1] http://www.poker770.com

Bibliography

[1] Fiedler, I., Wilcke, A.C.: Online Poker in the European Union. In: Gaming Law Review and Economics. (January/February 2012)

[2] Gilpin, A., Sandholm, T.: Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In: In International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). (2007)

[3] Gilpin, A., Sandholm, T.: A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In: proceedings of the 21st national conference on Artificial intelligence - Volume 2. AAAI'06, AAAI Press (2006) 1007–1013

[4] Billings, D., Davidson, A., Schaeffer, J., Szafron, D.: The challenge of poker. Artificial Intelligence **134** (2001) 2002

[5] Watson, I., Rubin, J.: CASPER: A Case-Based Poker-Bot. In: Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence. AI '08, Berlin, Heidelberg, Springer-Verlag (2008) 594–600

[6] Risk, N.A., Szafron, D.: Using counterfactual regret minimization to create competitive multiplayer poker agents. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1. AAMAS '10, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2010) 159–166

[7] Billings, D., Peña, L., Schaeffer, J., Szafron, D.: Using probabilistic knowledge and simulation to play poker. In: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence. AAAI '99/IAAI '99, Menlo Park, CA, USA, American Association for Artificial Intelligence (1999) 697–703

[8] Simm, J.: AI System for Online Poker. (2007)

[9] Smed, J., Knuutila, T., Hakonen, H.: Can we prevent collusion in multiplayer online games? In: Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence. SCAI 2006 (Oct 2006) 168–175