FABIAN MOIK

TITLE

Bachelor's Thesis

to achieve the university degree of

Bachelor of Science

Bachelors's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.mont. Pernkopf Franz

Institute for Signal Processing and Speech Communication

Head: Univ.-Prof. Dipl-Ing. Dr.techn. Some One

Graz, Oktober 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present bachelor‘s thesis.

Date	Signature

Abstract

This is a placeholder for the abstract.

Contents

Abstract iii

1. Introduction 1

2. Poker Basics 2

2.1. What is Poker . . . 2

2.2. No-Limit (NL) Texas Hold'em . . . 3

2.2.1. Poker Rules . . . 3

2.2.2. Betting . . . 5

2.2.3. Hand Rankings . . . 6

3. State-Of-The-Art 8

3.1. Knowledge-based Poker Agents . . . 8

3.2. Simulation-based Poker Agents . . . 9

3.3. Game-Theoretic Optimal Poker Agents . . . 10

3.4. Adaptive / Exploitive Poker Agents . . . 11

3.5. Bayesian Poker Agents and Evolutionary Algorithms . . . 11

3.5.1. Bayesian Poker Agents . . . 11

3.5.2. Evolutionary Algorithms 12

4. Implementation - Structure of the Bot 13

4.1. Architecture of Testbed and Neural Network Agents 14

4.1.1. Poker Testbed 14

4.1.2. Neural Network Agents 17

4.1.3. Key Components to Achieve High Level Poker Skill . 22

4.2. Betting Strategy 24

4.2.1. Preflop Strategy 24

4.2.2. Postflop Strategy 27

4.2.3. Betting Strategy 30

4.3. Training the NN-Agents 31

4.3.1. Evolution Phase 31

5. Experimental Results 34

6. Conclusion and Future Work 35

Bibliography 36

Appendices 39

A. Glossary of Poker Terms 39

List of Figures

4.1. High Level Architecture of Testbed 16
4.2. Architecture of the neural network with all the input features 18
4.3. The evolution process to train poker agents 33

List of Tables

2.1. Hand rankings for 5-card combinations (strongest to weakest) 7

1. Introduction

This is a placeholder for the introduction.

2. Poker Basics

2.1. What is Poker

In David Sklansky's [5] poker is being described as a term for a whole family of card games, which can be grouped into different types. In some game variants such as *Texas Hold'em or Omaha* the player with the highest ***hand rank*** [1] wins, in others such as *Razz or Badugi* the player with the lowest hand rank wins, and in yet others the ***pot*** is split between the player with the highest and the player with the lowest hand rank [5].

Among all poker variants there are different ***betting*** structures. Many poker formats offer ***limit*** variants of the game and ***no-limit*** variants. In limit games the ***bet size*** has an upper and lower limit, whereas in no-limit games the bet size is just restricted by the amount a player has currently left in his ***stack*** [5]. While bet sizes may not be restricted in no-limit games there are

[1]All poker specific terms mentioned the first time, appear in bold, italics throughout this bachelor's thesis. They are defined and briefly explained in the Appendix A.

specific rules in all poker variants that determine the minimum bet size at any given time in the game regardless of the variant of the game itself.

2.2. No-Limit (NL) Texas Hold'em

The focus of this work is drawn to the no-limit version of the Texas Hold'em poker variant. Not only is it the most widely played poker variant but also considered to be the most challenging form of poker [2]. While finding the optimal strategy for Texas Hold'em poker may be complex it has exceptionally simple rules [6].

2.2.1. Poker Rules

A Texas Hold'em ***hand*** begins in a stage called ***pre-flop***. In this stage every player at the table is dealt two cards. The cards are dealt face down and are exclusive to the player recieving the cards [6]. Each seat at the table has a special meaning throughout a poker hand. The seat (also called ***position***) which owns the ***dealer-button*** determines the two positions that have to place ***forced bets*** called the ***small blind*** and the ***big blind***. This dealer-button moves clockwise by one position once a whole poker hand is finished. The player directly to the left of the ***dealer***, the player currently holding the dealer-button, is forced to bet one small blind and the player two seats to the left of the dealer has to bet one big blind, the equivalent of two small blinds [1]. These bets serve the purpose of preventing players to wait for the best hand without any punishment.

In case of a Texas Hold'em poker tournament most often another type of forced bet called the ***ante*** is present in every hand of the tournament [7]. It usually is a percentage portion of the current big blind in the range of 10%-20% that has to be contributed to the pot by each player. In poker tournaments the small blind and the big blind are incremented after a fixed time interval to advance the game.

The pre-flop stage is concluded once each player has acted in the first betting round. The next stage, called the ***flop***, is introduced by three face up dealt cards to the ***community board*** and players once again have the chance to bet in this round. The third stage is called the ***turn*** and a fourth card is dealt to the community board. After concluding the third round of betting a fifth ***community card*** is dealt on the ***river*** and players are allowed to bet one last time. If there is more then one player left after this final round of betting the cards of all remaining players are revealed in a so-called ***showdown***. The best combination of two player's ***hole cards*** and three community cards decides the winner. If two players have the same hand rank, the pot is split between both of them [6].

2.2.2. Betting

Sophisticated betting strategies are the number one key component for succeeding in the game of NL Texas Hold'em poker. Choosing the optimal bet size in any given situation allows players to maximize their winnings and at the same time minimize their losses [1].

Each betting round is either opened by a bet or a ***check***. In a clockwise manner players may then decide to check or bet if there was no bet placed yet. If a bet has already been placed, the remaining players may either call the bet, ***raise*** the bet or ***fold*** to that bet. In NL Texas Hold'em the bet or raise amount is only limited by the remaining ***chips*** a player has [7].

Betting Options

- **Check / Call**: A check describes the action of staying in the hand without committing any money to the pot. To check a hand there must be no previous bets in the current betting round. By calling someone's bet a player wagers the amount of chips to equalize the amount of chips committed to the pot by each player.

- **Bet / Raise**: A bet describes the action of a player being first to put a certain amount of money into the pot. If there already exist a bet a player may still put more money into the pot then the previous bets did. This is referred to as a raise.

- **Fold**: Folding a hand means no longer being interest in the hand and not willing to put any more money into the pot. If however no bet has been placed in the current betting round yet, a player may rather check his hand instead of folding it.

[4, 7]

2.2.3. Hand Rankings

Texas Hold'em poker is a poker variant where the highest ranking 5-card combination wins [7]. To decide the winner of a poker hand the best possible 5-card combination of each remaining player is compared against each other [3]. The ranking system of 5-card combinations can be found in Table 2.1.

Hand rank and sample hand	Description
Royal Flush A♠ K♠ Q♠ J♠ 10♠	Highest ranking Flush plus Straight
Straight Flush J♣ 10♣ 9♣ 8♣ 7♣	5 cards of same suit and in sequence, without the Ace
Four of a Kind Q♣ Q♥ Q♦ Q♠ 2♠	Four matching cards of same rank
Full House Q♣ Q♥ Q♦ 2♣ 2♠	Three of a kind and one pair
Flush 7♣ Q♣ 9♣ 2♣ K♣	5 cards of same suit, not in sequence
Straight 7♣ 8♦ 9♥ 10♣ J♣	5 cards in sequence without matching suits
Three of a Kind Q♣ Q♦ Q♥ 8♥ 2♣	Three cards of the same rank, two unmatched cards
Two Pair 7♣ 7♦ 9♦ 8♥ 8♣	Two pairs of two matched cards
One Pair 7♣ 10♦ 9♦ 8♥ 8♣	Two cards of the same rank, three unmatched cards
High Card 7♣ 10♦ 9♦ A♥ Q♣	No matches between 5 cards. The highest card counts.

Table 2.1.: Hand rankings for 5-card combinations (strongest to weakest)

3. State-Of-The-Art

The game of poker has many interesting properties which proved to offer a challenging test environment for artificial intelligence research. Over the last decade scientists and researchers have studied the game from a game theoretical view to find optimal solutions but also tried a number of machine learning algorithms and artificial intelligence systems on the game [12, 15]. A lot of research though focused on simplified versions of the game because these variants of poker are easier to analyze but still offer demonstrations of game theoretical principles [15]. Nonetheless over the past few years, abstract versions of poker have been used to first successfully train algorithms on this simple version of the game and then transfer the obtained knowledge to a full version of the game [12, 15].

3.1. Knowledge-based Poker Agents

Knowledge-based systems can be broken into two categories, namely **rule-based expert systems** and **formula-based methods**. In general knowledge-based systems require the knowledge of an expert player to design the system [4]. A *rule-based expert system* in its simplest form is a sequence of if-else statements for frequently occurring scenarios of the game which determine the desirable ***betting action***. More advanced human players describe their poker hands in a very similar way when they break them down in a discussion [1]. *Formula-based methods* on the other hand try to generalize the problem by defining a formula that takes a set of inputs and outputs a so-called ***probability triple*** upon which a betting decision is made. Inputs to the formula may describe important information about the current game state and can be weighted [4].

While rule-based expert systems may yield reasonably good results in the first stage of a poker hand (pre-flop), they fail to shine in later stages of the game because they become increasingly difficult to maintain with additional and sometimes even conflicting information added at each stage. Furthermore static strategies are prone to exploitation and therefore not competitive with other approaches [4].

3.2. Simulation-based Poker Agents

In general simulation based methods rely on repeatedly simulating an outcome in order to approximate the resulting expected value of an action [1]. A frequently used simulation method applied to the game of poker is the so called *Monte-Carlo simulation* or *Monte-Carlo Tree Search*, which is a procedure that searches the game tree by sampling the possible choices in a game state and simulates the action taken to the bottom of the tree. By repeating this procedure a robust expectation value can be computed [4]. In [10] Billings *et. al* describe the technique of *selective sampling* which uses information about the opponents to bias the selection of possible card combinations a player may hold. With this modification to the sampling algorithm they were able to create a more dynamic betting strategy due to the gain of valuable information about opponants.

3.3. Game-Theoretic Optimal Poker Agents

The currently best performing poker-playing programs are approximating a Nash equilibrium [9]. In game theory a Nash equilibrium describes a state of a game where no player can find an action that would yield a better outcome than the suggested equilibrium action, given that all other players also choose to take the suggested action [8].

This strategy has proven to achieve great results in zero-sum games, like NL ***heads-up*** poker [1]. Most successful heads-up poker bots use an abstract version of the poker variant in which they approximate the Nash equilibrium and then transfer the decision made in the abstract version to the real version of the game. One of the most successful algorithm in approximating the Nash equilibrium in an abstract version of the game is called *Counterfactual Regret Minimization*. The top three poker bots in the *Annual Computer Poker*

Competition (ACPC) 2016 used a variant of the Counterfactual Regret (CFR) algorithm to defeat their competitors [9].

3.4. Adaptive / Exploitive Poker Agents

Nash equilibrium approaches and other static poker strategies are vulnerable to exploitation [1]. Adaptive strategies try to tackle this problem by quickly adapting to the playing-style of the opponents and exploiting their weaknesses. Two algorithms, namely the *Miximax* and *Miximix* algorithm achieve this by searching an adaptive imperfect information game tree. Decisions are then made by considering a *randomized mixed strategy* associated with the decision node of the searched tree [6].

3.5. Bayesian Poker Agents and Evolutionary Algorithms

3.5.1. Bayesian Poker Agents

A Bayesian network is a probabilistic graphical model. Each node in the directed acyclic graph represents a random variable and edges between nodes represent the conditional dependencies of variables. Nodes are associated with a probability function which returns the conditional probability values based on their parent's values. A probability distribution over the random variables can be retrieved by propagating the probabilities of initialized nodes throughout the network [4].

Compared to other poker playing agents, bayesian agents performed badly in the AAAI Computer Poker Competitions in the last years. There is still a lot of room for improvement in Bayesian based networks and further research in this field has to be done to stand a chance in upcoming competitions [4].

3.5.2. Evolutionary Algorithms

Yet to come, follows when EA chapter is finished

4. Implementation - Structure of the Bot

In this chapter the architecture of both the test environment and the poker playing agents is described and the learning process of the poker agents is explained. There are some key components that distinguish a strong poker player from a weak one. A well defined *betting strategy* both attuned to the mathematical principles such as *hand strength (HS)* and *hand potential (HP)* and to the interpretation of opponent's tendencies decides on the profitability of a poker player over the ***long run*** [11]. This chapter describes a possible way to implement and apply these concepts to create a poker playing agent, capable of understanding the situation on the table and assessing its hand strength versus opponents.

4.1. Architecture of Testbed and Neural Network Agents

Although there are a handful open source poker testbeds on the internet, none of them has proven to be suitable to achieve the goal of this thesis. Some of them lack the needed flexibility to extract all necessary information for the bot out of the current game state, others are not designed to run tournament poker simulations but only ***cash games*** or other poker variants. Therefore in this thesis a testbed was crafted, which can be arbitrarily modified to the needs of the user. However, the main focus of this thesis was on the creation of a test environment that can simulate millions of poker tournaments in as little time as possible. At the same time it should provide our poker agents with all the needed information about the current game state and opponents at the table. The poker testbed was written in *C++* due to little performance overhead at runtime and its speed and efficiency.

4.1.1. Poker Testbed

The basic structure of the poker test environment, referred to as *testbed* in this context consists of following components:

- ▷ Game object
- ▷ Table object
- ▷ Dealer object
- ▷ Player object
- ▷ Artificial Intelligence (AI) object
- ▷ Deck object
- ▷ Card object
- ▷ Rules object

To run an arbitrary number of simulated tournaments with one generation of agents, a *game* object is initialized. The *rules* for the game, including the ***Buy-In***, the total number of players, the maximum number of players per table and the ***blind structure***, are then passed to the game object and all participating *players* are added to the game.

Depending on the number of players participating in the tournament a number of *table* objects are created. The game object then distributes all players across the tables and places one *dealer* on each table. Dealers are not participating in the tournament but their job is it to deal cards to the players, apply forced bets (*Small Blind*, *Big Blind* and *Antes*), deal community cards, determine the winning player for each hand and split the pot accordingly. Each dealer holds a *deck* object which consists of 52 unique cards, 13 cards for each suit. The *cards* themselves are also objects and contain the information about their ***suit*** and the card's value. They also offer some convenience functions for representing the card's value in a readable format for the human.

Each player holds an *AI* object and is assigned a unique ID. The dealer tells the player when to act and provides him with all the public information

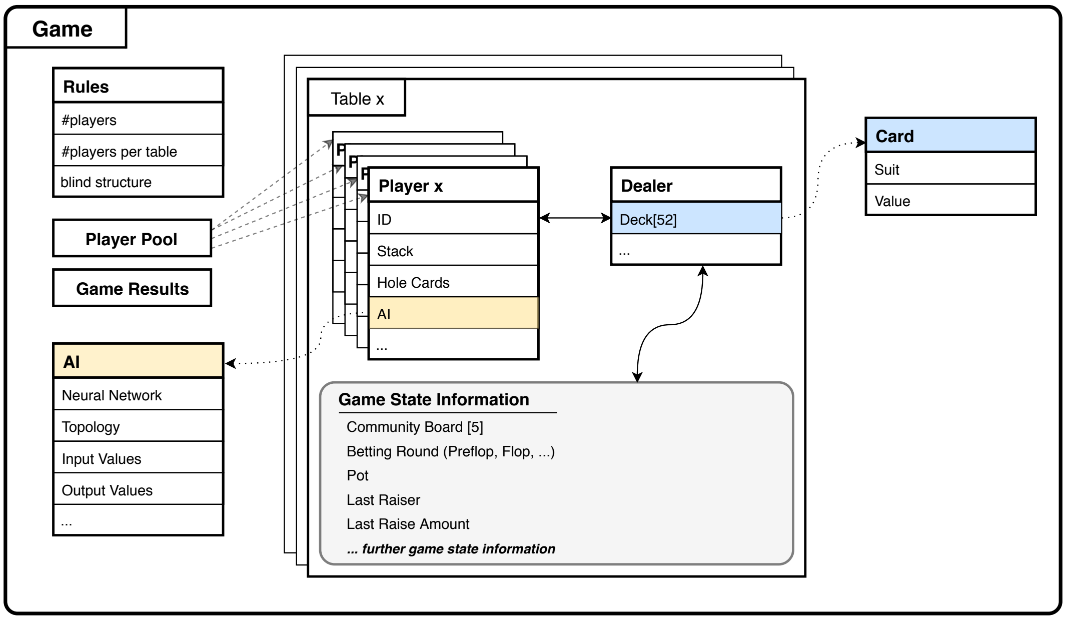

Figure 4.1.: High Level Architecture of Testbed.

about the game state. The player's AI then decides on the basis of this information how it would act in this situation and the player executes this action. In case of a betting action the player selects an appropriate bet size according to a betting strategy and places it on the table. An in depth look into the architecture of the bot is provided in subsection 4.1.2.

Figure 4.1 shows the high level architecture of the described testbed. Boxes represent objects, list items within the box represent properties of the corresponding object. Bidirectional arrows indicate that information is exchanged in both directions between two objects. Dotted arrows zoom in on a property of an object. Rounded boxes represent a collection of properties of an object.

4.1.2. Neural Network Agents

In the conducted test series of simulating hundreds of tournaments in a single population of No-Limit Texas Hold'em poker agents, evolving neural networks were used to train the agents on the game of NL Hold'em poker. An agent in this population is represented by the *player object*, described in 4.1. The player object holds an *AI object*, that returns the desired action to be taken, given the public information about the current game state. This public game state information is provided by the *dealer object*.

Structure of the Neural Network

The agents to be examined are implemented as fully connected feed-forward neural networks with a network topology of 24-14-3, which corresponds to 24 *input neurons* in the input layer, 14 *hidden neurons* in the hidden layer and three *output neurons* in the output layer [13]. For the hidden layer a **sigmoid** activation function is used and a **softmax** activation is performed at the output layer. The three output neurons represent the so called **probability triple** (f, c, r), which specifies the probability distribution for the actions *fold, check/call* or *bet/raise* at the current state of the game [4]. The full architecture of the neural network with all its input features can be seen in Figure 4.2. The selection of input features was strongly influence by the work of Nicolai in [14].

The final structure has not yet been determined (depends on opponent modeling module)

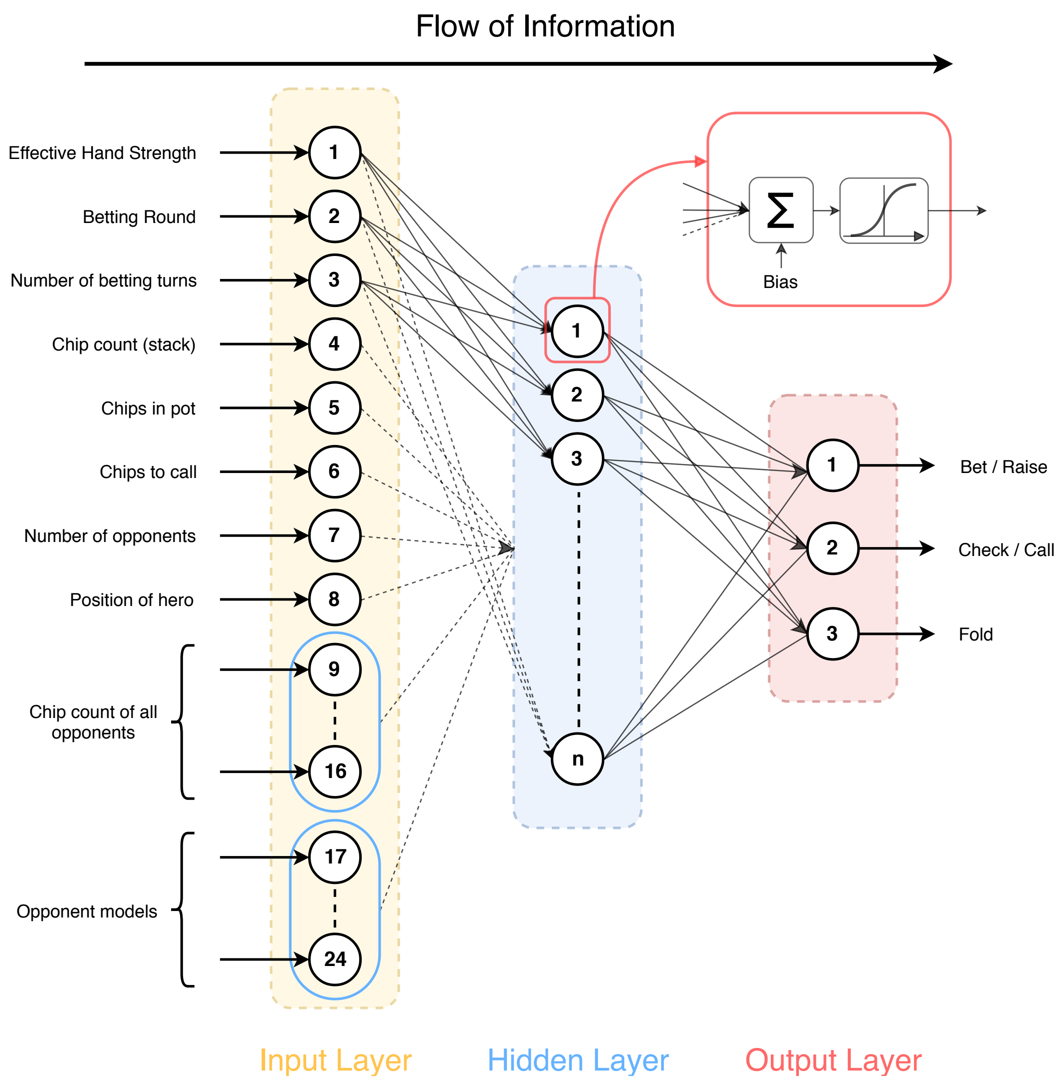

Figure 4.2.: Architecture of the neural network with all the input features

Effective Hand Strength (EHS)

The EHS is an indicator for how likely a hand is winning at showdown considering the current hand strength but also the positive and negative hand potential [14]. In conjunction with other components the EHS should be used to help selecting a suitable betting action. A more detailed description of EHS can be found in subsection 4.2.2.

Betting round

The second input for the neural network is the current betting round in the game. This can be PREFLOP, FLOP, TURN or RIVER. The betting round is an important property because a good preflop strategy varies from a good postflop strategy and therefore this property should not be withheld from the neural network.

Number of betting turns

The next input to the neural network is the current turn of betting. A ***betting turn*** starts with the first player betting chips into the pot and is concluded once the action returns to this exact player. There can be multiple betting turns in each round of the game.

Chip count

The fourth feature is the number of chips the acting agent has. This represents the stack of the player.

Chips in pot

The fifth feature is the number of chips already in the pot, including all chips of previous betting rounds plus the number of chips betted by other agents in the current betting round. An agent may not be able to win all the chips in the pot if his chip count (stack) is smaller than a bet of his opponents in the current round. Hence the value of this feature is the effective pot size an agent can actually win when his hand is the strongest at showdown [14].

Chips to call

The sixth feature is the number of chips an agent has to match in order to be allowed to continue in the hand. The ratio between the amount of chips a player has to bet to stay in the hand and the amount of chips in the pot is called ***pot odds***. This is a commonly used tool in the poker community to calculate the needed winning percentage of an agent's hand to make the call a profitable play, irrespective of other factors [14].

Number of opponents

This feature represents the number of opponents still involved in the hand.

Position of hero

This feature is the relative position of the agent to the dealer. The dealer position is the most valuable position in poker because the action of only two more players will follow in an unopened pot. In general it is desirable to be in a ***late position*** because many players have already acted before you and hence more information is available for the agent in late position [14].

Chip count of all opponents

In a ***9-handed*** multi table tournament there are at most 9 players on one table. Therefore this feature has 8 input nodes, one for each opponent on the table. The input is the number of chips an opponent has. Late in a tournament it will occur that hands are not always played 9-handed but with less then 9 players per hand. For this case the input for empty seats on the table is set to unknown and will be recognized by the neural network as an empty seat. The first node of this 8 nodes is always the opponent directly to the left of the agent, the second node is the opponent two seats to the left of the agent [14].

Opponent model of all opponents

YET TO COME! → opponent modeling module needs to be finished first

4.1.3. Key Components to Achieve High Level Poker Skill

Billings et al. [15] described five main requirements a poker playing agent needs to fulfill in order to be competitive with the best players in the world. All these components depend on each other and need to be adjusted when a certain situation requires it. The five requirements mentioned by Billings are:

- ▷ Hand Strength
- ▷ Hand Potential
- ▷ Bluffing
- ▷ Unpredictability
- ▷ Opponent Modeling

Hand strength and **hand potential** are explained in detail in subsection 4.2.1. In short HS and HP combined asses the relative hand strength against opponents, ideally considering opponent-specific factors that would influence the probability of winning a hand [15].

To achieve this requirement a slightly different approach than recommended in [15, p. 208] was implemented in this study. Instead of considering opponent-specific tendencies while calculating the hand strength and hand potential, the effective hand strength was calculated under the assumption of playing against a random player, the reason being that opponent's tendencies are later given as input to the neural network which should be able to convert this information into meaning.

Opponent modeling as described by Billings et al. [15, p. 208] tries to accumulate information about the playing style of an opponent. Later this information is applied to hand strength calculations and used to create likely probability distributions of opponent's hand ranges.

Again in our study opponent models are not exactly used in the same way as described but rather represent key opponent-specific tendencies which do not directly influence any calculations but rather serve as additional input information for the neural network, to find an optimal betting strategy against different opponent types.

Unpredictability can be achieved by altering our strategy in a given situation. This means that a player should sometimes play differently in a similar situation, to not allow opponents to generate a precise model of our strategy [15].

This component is achieved in our study by following certain probability distributions when it comes to bet sizes and betting actions. A detailed description the betting strategy follows in subsection 4.2.3.

Bluffing is the last requirement mentioned by Billings et al. [15] and serves the purpose of sometimes winning pots with weak hands. By sometimes bluffing with weak hands we make opponents doubt our hand strength and later win more money on our strong hands against them.

The bluffing component was not explicitly design in our study, but ideally the neural network should discover the concept of bluffing on its own, by interpreting the given opponent models and its relative hand strength.

4.2. Betting Strategy

Billings *et al.* [15, p. 210] emphasis that betting strategies for pre-flop play and post-flop play are considerably different and that “a relatively simple expert system is sufficient for competent play” pre-flop. Nonetheless for the conducted study no expert system for pre-flop betting decisions was used but an evolving neural network was provided with public game state information for all betting rounds. While sharing the same neural network for both pre-flop and post-flop play there is one significant difference in calculating the HS for pre-flop decisions compared to post-flop calculations.

4.2.1. Preflop Strategy

The state space in the pre-flop stage of a poker hand is relatively small compared to the post-flop state space. In total there are $\binom{52}{2} = 1352$ initial hole card combinations pre-flop, but this translates to only 169 distinct hand types because some hands have the same strength pre-flop but not post-flop. For example A♥ K♥ and A♦ K♦ have the same strength pre-flop and therefore are categorized into one distinct hand type, namely Ace King ***suited*** (AKs) [12].

For evaluating the hand strength of one agent against N opponents a lookup table was created. It consists of $169 * 8 = 1352$ entries, 169 distinct hand types played against one to eight opponents. A *Monte-Carlo Simulation* of one million poker hands was performed for each distinct hand type against each possible number of opponents. During those one million hands the agent's hole cards were ranked against all opponents ' hole cards and all wins, ties and losses for the agent were counted. An approximation of the expected win percentage was calculated with following formula:

$$HS = \frac{(WINS + \frac{TIES}{2})}{WINS + TIES + LOSSES} \tag{4.1}$$

The hand strength (HS in the formula) represents the chance of a hand beating a random hand under the assumption of seeing all 5 community cards and going to showdown. This approach does not take opponent models into consideration [12].

Hand Ranker

A hand ranker as the name suggest, ranks a given poker hand based on its relative value by taking 5 to 7 cards as input and returning a number. The higher the number returned by the hand ranker, the better the hand

[16]. There exists a variety of open source hand ranking algorithms on the internet but only a few of them were able to hold their own against the competition over the years. The reason why only a hand full of poker hand ranking algorithms are used nowadays is the need for speed. Writing a simple hand ranker is rather easy, but writing an efficient one that is able to rank millions of hands every second is not so trivial. Among the most famous hand ranking algorithms one especially excelled due to its speed. It was created by the *TwoPlusTwo (TPT)* community and uses a look up table with 32487834 entries which translates to a file size of approximately 250MB [16]. The TwoPlusTwo poker hand ranker is one of the fastest of its kind to date. Not only is it extremely fast but also able to rank 5-card, 6-card and 7-card poker hands. This becomes very handy when evaluating hands pre-flop, on the flop and on the turn. For this reason the TPT hand ranker was used in multiple hand strength calculations throughout this study to determine if a given hand beats an other hand.

Hand Strength

While a hand ranker assigns a value to a given hand, it can only tell if a certain hand is better than other hands, and not how likely this hand currently is to win against a random hand. This is where the hand strength calculation comes into play. A simple approach of achieving this is to enumerate all possible remaining hand combinations an opponent could hold and count the number of wins, ties and losses versus the agent's hand.

Using Formula 4.1 one gets the probability that the current hand beats a random opponent's hand at the current round of the poker game. This value is also called the *raw hand strength (RHS)*. While this approximation in combination with Monte-Carlo simulations might be sufficiently accurate for pre-flop hand strength assessment it is certainly insufficient for post-flop situations, when viewed in a vacuum. Raw hand strength calculations disregard the future potential of a hand and only represent the probability of a hand beating a random hand if there were no cards to come [12]. To overcome this obstacle the *Effective Hand Strength* algorithm was conceived by Billings et al. and first published in [11].

4.2.2. Postflop Strategy

For evaluating the strength of a hand after the pre-flop stage, it is important to not only consider the raw hand strength but also the positive and negative hand potential. Both metrics combined form the so-called *effective hand strength* [3].

Effective Hand Strength

The raw hand strength combined with the positive and negative hand potentials yield a measurement for the relative hand strength against an opponent competing in the poker hand. In general the effective hand strength can be represented by following formula, as Billings et al. described it in [15, p. 216]:

$$EHS = HS \times (1 - NPot) + (1 - HS) \times PPot \tag{4.2}$$

NPot represents the negative pot potential, which stands for the probability of falling behind opponents when we currently have the best hand. Conversely *PPot* stands for the positive hand potential, the probability of improving the hand when currently behind versus opponents. The EHS calculation can be generalized for *n* opponents by simply raising the HS to the power of *number of opponents* [11]. This yields:

$$EHS = HS^n \times (1 - NPot) + (1 - HS^n) \times PPot \tag{4.3}$$

Although Billings et al. do not recommend to generalize the EHS calculation we decided to do so in our study because the accumulated information about tendencies of opponents was not directly used to influence hand strength calculations but rather serve as additional input to the neural network which in turn should learn to apply these models on its own.

Hand Potential

Hand potential calculations are used to account for future cards to come in a poker hand and to assess the possible impact of these cards on the current raw hand strength [15][1]. Assuming the current poker hand is in

[1]More details on hand potential calculations and associated algorithms can be found in [15, p. 216-218].

the flop-stage, with three cards already dealt to the community board, then there are $\binom{45}{2} = 990$ possible combinations of future turn and river cards for every possible hand an opponent might hold. That means if one would like to calculate the hand potential of a current hand against a random opponent that would equate to $\binom{47}{2} \times \binom{45}{2} = 1081 \times 990 = 1070190$ calculations for this situation.

Because such a number of calculations is too expensive for our training algorithm, the effective hand strength calculation was modified to approximate the EHS.

Effective Hand Strength Approximation

YET TO COME! $\rightarrow$ the approximation algorithm is implemented but not yet described.

4.2.3. Betting Strategy

YET TO COME! $\rightarrow$ the betting strategy is adopted from [14, p. 53 - p. 57]. It is fully implemented but some parameters are still varied, so the description follows when final results are ready.

4.3. Training the NN-Agents

Unlike other games such as Chess, Checkers or Go, Poker is a game of imperfect information. This means, that a player can not see all relevant information at a given game state at all times. Because the game of Poker involves hidden information and deception, players must be willing to take risks based on the information they have at the current state of the game [17]. Unlike other methods, which reduce the large decision space of No-Limit Hold'em by transforming the game into a smaller abstract version in which they then approximate the optimal path and execute the result in the original game via a translation method, Nicolai and Hilderman [17] introduced a method, in which evolutionary algorithms can be used to train neural network agents to achieve reasonably good results in the game of No-Limit Texas Hold'em poker. The proposed algorithm in their work is used as a guide in this thesis to train neural network agents through iterative play over a large number of generations by mimicking natural evolution [17].

4.3.1. Evolution Phase

Evolutionary algorithms try to mimic the evolution process as it is know in biology. Agents in a population are given a task and try to achieve the best results possible. The performance of agents is then evaluated and the best among the population are chosen to reproduce offsprings for the next generation. Offsprings are almost identical to their parents, but with slight changes to their genetic material. These offsprings together with the best agents from the previous generation are then again given the same task and the whole process repeats. [14, 17].

In case of our neural network agents, the population of the first generation consists of randomly weighted neural network agents. They are competing in a number of tournaments against each other and some pre-defined benchmark agents. The best agents of the first generation are then chosen to reproduce offsprings. Together with the best agents of the previous generation, all offsprings are then competing in another series of tournaments until after N generations the process is stopped.

Figure 4.3 shows the implementation of simulating a number of tournaments for each generation and selecting a number of elite players per generation to evolve the neural network agents.

This section is still in work. Figure 4.3 will be explained in more detail. Additionally a code listing of *evolving agents* will follow. The method of combining two parents to form offsprings will be explained. The evaluation of agents in a generation by using a certain fitness function will also be explained in a subsection, once the implementation has produced the final results. Furthermore the concept of a **Hall of Fame** will be explained and how it serves as a genetic memory for an evolutionary system.

```
void RunEvolution(int num_generations, int num_players, int num_elite_kept, int
    num_tournaments) {

    std::vector<Player*> players;
    std::vector<Player*> hall_of_fame;

    // Creating players for the first generation
    for (int i = 0; i < num_players; i++) {
        Player *player;
        AIOwn *ai = new AIOwn();
        player = new Player(ai);
        player->setID(i);
        players.push_back(player);
    }

    // Initializing Hall of Fame
    InitializeHallOfFameWithOwnAi(players, hall_of_fame);

    for (int i = 0; i < num_generations; i++) {
        // Elite players per generation
        std::vector<Player*> elite;

        elite = RunOneGeneration(players, rules, num_elite_kept, num_tournaments,
            hall_of_fame);

        // Evolve agents
        std::vector<Player*> players_to_evolve(players.begin(), players.begin() +
            num_players);
        EvolvePlayers(players_to_evolve, elite);

        generation++;
    }
}
```

Figure 4.3.: The evolution process to train poker agents

5. Experimental Results

This is a placeholder for the results.

6. Conclusion and Future Work

This is a placeholder for the conclusion and future work.

Bibliography

[1] N. Passos. *Poker Learner: Reinforcement Learning Applied to Texas Hold'em Poker*, Master's thesis, Faculdade de Engenharia da Universidade do Porto, Portugal, 2011. https://paginas.fe.up.pt/~ei08029/Master Thesis - Nuno Passos.pdf

[2] J. Schaeffer, D. Billings, L. Peña, and D. Szafron. *Learning to Play Strong Poker*. ICML-99, Proceedings of the 16th International Conference on Machine Learning, 1999. http://poker.cs.ualberta.ca/publications/ICML99.pdf

[3] L. Peña. *Probabilities and simulations in poker.* Mather's thesis, Department of Computing Science, University of Alberta, 1999.

[4] J. Rubin, and I. Watson. *Computer poker: A review.* Artificial Intelligence, 175(5-6):958-987, 2011.

[5] D. Sklansky. *The Theory of Poker.* Two Plus Two Publishing, 2005.

[6] D. Billings. *Algorithms and Assessment in Computer Poker* Ph.D. Dissertation, University of Alberta, 2006.

[7] R. D. Harroch, and L. Krieger. *Poker For Dummies.* John Wiley & Sons, 1st Edition, 2000.

[8] M. J. Osborne. *A Course in Game Theory.* The MIT Press, Cambridge, Massachusetts, London, England, 2014.

[9] V. Lisy, and M. Bowling. *Equilibrium Approximation Quality of Current No-Limit Poker Bots.* arXiv preprint arXiv:1612.07547, 2017.

[10] D. Billings, L. Peñe, J. Schaeffer, D. Szafron. *Using Probabilistic Knowledge and Simulation to Play Poker.* In 16th National Conference on Artificial Intelligence, pp. 697-703, 1999.

[11] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. *Opponent Modeling in Poker.* Proc. AAAI-98, Madison, WI, pp. 493-499, 1998.

[12] A. Davidson. *Opponent modeling in poker: Learning and acting in a hostile and uncertain environment.* Master's thesis, University of Alberta, 2002.

[13] G. Nicolai, and R. J. Hilderman. *No-Limit Texas Hold'em Poker Agents Created with Evolutionary Neural Networks.* CIG-2009, IEEE Symposium on Computational Intelligence and Games, pp. 125-131, 2009.

[14] G. Nicolai. *Evolutionary methods for learning no-limit Texas hold'em poker.* Master's thesis, University of Regina, 2008.

[15] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. *The Challenge of Poker* Artificial Intelligence 134, pp. 201-240, 2002.

[16] L. F. Teofilo. *Estimating the Probability of Winning for Texas Hold'em Poker Agents*. Proceedings 6th Doctoral Symposium on Informatics Engineering, pp 129-140. Porto, Portugal.

[17] G. Nicolai, and R. J. Hilderman. *Algorithms for Evolving No-Limit Texas Hold?em Poker Playing Agents.* In Proceedings of the International Conference on Evolutionary Computation, 2010.

A. Glossary of Poker Terms

This appendix contains definitions and brief explanations of all used poker terms in this bachelor's thesis. A full glossary can also be found on the web, following the link: https://en.wikipedia.org/wiki/Glossary_of_poker_terms

- ***9-handed***: A game of poker where 9 players are sitting at the table.
- ***Ante***: A forced bet in some variants of poker (especially in tournament poker) that every player has to pay in order to receive cards.
- ***Bet***: A bet describes the amount of chips a player puts into the pot in a betting round when the pot it unopened.
- ***Bet size***: The amount of chips wagered in a betting action.
- ***Betting action***: Describes the type of action in a betting situation. For example: Bet, 2-Bet, 3-Bet, 4-bet.
- ***Betting turn***: needed?
- ***Big blind***: A forced bet that the player two positions to the left of the dealer has to pay before receiving his cards. The Big blind is usually twice the size of the small blind.

- ***Blind structure***: This is a poker tournament specific term, which describes the temporal structure of blinds. Blinds are usually increased in fixed time intervals.
- ***Blinds***: Force bets that are split into small blind and big blind. The player immediately to the left of the dealer has to pay the small blind, the player two positions to the left of the dealer has to pay the big blind in order to receive cards.
- ***Buy-In***: Describes the amount of money payed to enter a tournament and gain chips.
- ***Call***: Matching a bet made by other players in the current betting round.
- ***Cash game***: It is a poker variant where the chips at the same time represent real money value. The blinds are staying the same over time.
- ***Check***: The action of staying in a hand without committing any money to the pot. This is only possible if the betting round is unopened.
- ***Community board***: Describes the shared cards on the table as an entity.
- ***Community card***: Face up dealt cards on the table, shared between all players.
- ***Dealer***: Either a person on the table dealing cards to players or the position on the table holding the dealer button.
- ***Dealer-button***: An object marking the seat on the table that receives the last card dealt pre-flop.
- ***Flop***: The first three cards dealt face up to the community board. It is followed by the second betting round.
- ***Fold***: The action of laying down/ giving up a hand.

- ***Forced bets***: Bets that have to be payed in order to receive cards pre-flop. This includes the small blind, the big blind and antes.
- ***Hand***: The cards dealt to a player.
- ***Hand rank***: The relative strength of a hand at showdown. In Texas Hold'em poker the best 5-card combination defines the hand rank.
- ***Heads-up***: When only two players are competing in a hand.
- ***Hole cards***: The two face down cards dealt to a player pre-flop.
- ***Late position***: Describes the two players sitting on the dealer position and one seat to the right of the dealer (the so-called "Cut-Off" position").
- ***Limit poker***: A poker variant where there are certain minimum and maximum limits on bet sizes.
- ***Long run***: Describes a lengthy time period.
- ***No-Limit poker***: A poker variant where there are no minimum or maximum limits on bet sizes.
- ***Pre-flop***: A betting round before the flop. In the pre-flop stage of a hand player are being dealt cards and have the chance to bet.
- ***Position***: The position on the table relative to the dealer seat. Player close to the left of the dealer are in early position and players close to the right of the dealer are in late position.
- ***Pot***: The accumulated amount of chips wagered by all players combined, that is awarded to the winner of the hand.
- ***Pot odds***: The ratio between the size of the pot and the amount of chips needed to call.
- ***Probability triple***: Describes a probability distribution across the three

possible betting options *bet/raise, check/call, fold*.

- ***Raise***: Raising a bet means first matching a bet made by an other player and increase that bet by a certain amount.
- ***River***: The fifth card dealt face up to the community board. It is followed by the fourth and final betting round.
- ***Showdown***: If more than one player remains after the last betting round (River), the player's hands are exposed and a winner is determined.
- ***Small blind***: A forced bet that the player immediately to the left of the dealer has to pay before receiving his cards.
- ***Stack***: The total amount of chips or money a player has currently available to play on the table.
- ***Suited***: When both hole cards share the same suit
- ***Turn***: The fourth card dealt face up to the community board. It is followed by the third betting round.