FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Poker Learner: Reinforcement Learning Applied to Texas Hold'em Poker

Nuno Miguel da Silva Passos

Master in Informatics and Computer Engineering

Supervisor: Luís Paulo Reis (Ph.D.)

Co-Supervisor: Luís Teófilo (Ms. C.)

18th July, 2011

Poker Learner: Reinforcement Learning Applied to Texas Hold'em Poker

Nuno Miguel da Silva Passos

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: António Coelho (Ph. D.)

External Examiner: Luís Correia (Ph. D.)

Supervisor: Luís Paulo Reis (Ph. D.)

18th July, 2011

Resumo

Os jogos sempre foram uma área de interesse para a inteligência artificial, especialmente devido às suas regras simples e bem definidas e a necessidade de estratégias complexas para garantir a vitória. Vários investigadores da área das ciências da computação dedicaram o seu tempo no estudo de jogos como o Xadrez ou Damas, obtendo resultados notáveis, tendo desenvolvido jogadores artificiais capazes de derrotar os melhores jogadores humanos.

Na última década, o Poker tornou-se num fenómeno de popularidade crescente a nível mundial, comprovado pelo aumento consideravel do número de jogadores nos casinos *online*, tornando o Poker numa indústria bastante rentável. Na área de investigação da inteligência artificial, o Poker também tornou-se igualmente num domínio interessante devido à natureza estocástica e de informação incompleta do jogo. Por estes motivos, Poker apresenta-se como uma boa plataforma para desenvolver e testar soluções para desafios presentes na área de inteligência artificial, como o tratamento de informação incompleta do ambiente e a possibilidade de existirem elementos aleatórios no mesmo. Já foram seguidas diversas abordagens para conseguir desenvolver um jogador artificial perfeito mas, até à data, tal ainda não foi possível. No entanto, já foram feitos avanços significativos nesse sentido, nomeadamente na área de estimativa de sucesso de mãos, onde é calculada a qualidade da mão baseada na força da combinação das cartas juntamente com a probabilidade de a presente mão poder resultar numa mão melhor. Outra área desenvolvida é a de modelação de oponentes, onde o objectivo é identificar o tipo de oponente e adaptar a maneira de jogar do agente para melhorar o desempenho contra os adversários.

O objectivo deste projecto de dissertação consiste no desenvolvimento de uma nova abordagem para concepção de agentes de Poker, baseada em aprendizagem por reforço. Enquanto a maior parte dos jogadores artificiais segue regras estáticas definidas pelos seus criadores, o agente desenvolvido neste projecto constroi a sua estratégia baseado na experiencia adquirida durante os jogos. Esta é também uma boa oportunidade de verificar e testar qual a melhor maneira de descrever o estado do ambiente num jogo de Poker.

Para atingir estes objectivos foram desenvolvidos quatro agentes com estruturas e funções semelhantes. Durante o jogo os agentes verificam o estado de ambiente e tomam decisões que

tragam a melhor recompensa final. No final do jogo, a recompensa de cada acção é actualizada de acordo com os resultados obtidos.

No final do projecto foram obtidos quatro agentes capazes de, sem conhecimento prévio sobre a força de cada mão, aprender a jogar e conseguir ganhar a agentes de um nível básico/intermédio numa mesa com número variável de jogadores, com diferentes estilos de jogo. Com estes resultados é possível concluir que Aprendizagem por Reforço é uma abordagem viável ao jogo de Poker.

Abstract

Games have always been an area of interest for artificial intelligence, especially because of its simple and well defined rules and the need of complex strategies to assure victory. Several researchers from the area of computer science have dedicated their time to the study of games like Chess or Checkers, obtaining notoriety by developing an artificial player capable of defeating the best human players.

In the last decade, Poker has become a worldwide phenomenon of growing popularity, proved by the considerable raise in the number of players on online casinos, making Poker a very profitable industry. For the artificial intelligence research field, Poker has also become an interesting domain due to the game's stochastic and incomplete information nature. Because of these two motives, Poker presents itself as a good platform to develop and test solutions for challenges in the artificial intelligence field, like the treatment of the environment's incomplete information and the possibility of random events present in the same environment. Several approaches were made to develop the perfect artificial player, but currently that wasn't possible. However, there have been several advances towards that goal, especialy on the hand success estimate, where the quality of the hand is calculated based on the sthrenght of the card combination together with the probability of a better hand resulting of the current hand. Other area developed is opponent modeling, where the objective is to identify the type of opponent and adapt the agent's playing style to improve its performance against those opponents.

The objective of this dissertation project consists in the development of a new approach to the building of Poker agents based on reinforcement learning. While most artificial players follow static rules defined by their developers, the agent developed on this project builds its own strategy based on the experience that it acquires while playing the game. This is also a good opportunity to ascertain and test the best way to describe the state of the environment on a Poker game.

At the end of the project, there were developed four agents capable of, without previous knowledge of hand ranking, learn how to play and manage to win against agents of a basic/intermediate level, in a table with any number of players, with different playing styles. With these results it is possible to conclude that reinforcement learning is a viable approach to the game of Poker.

Acknowledgements

First of all, I would like to thank to Prof. Dr. Luís Paulo Reis for proposing this dissertation project and for its supervision of all the work done.

Second, to Luís Teofilo, who helped me from the start by providing me all the necessary tools. He also gave me plenty of ideas to enhance my work.

Finally, but not least important, to my family and friends for their support, patience and encouragement during the project duration, and even before that.

Nuno Miguel da Silva Passos

Index

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
API	Application Programming Interface
CPRG	Computer Poker Research Group
LIACC	Artificial Intelligence and Computer Science Laboratory

Chapter 1

1.Introduction

1.1 Context

> "Artificial Intelligence (AI) is the science and engineering of making intelligent machines [1]".

This sentence was used by John McCarty to announce the new sub-field of research in computer science, sentence that was accepted in a conference at the Dartmouth College in the summer of 1956 [2].

The idea of intelligent machines is not new, since it appeared in several ancient Greek myths, such as the bronze robot of Hephaestus or Pygmalion's Galatea. There are also newer references in science fiction such as Mary Shelley's Frankenstein [3] or Karel Čapek's Rossum's Universal Robots.

The common idea in all these references is the presence of the concept of Strong AI, also known as General AI. This type of artificial intelligence first objective was to create a machine capable of matching a human mind, but due to unforeseen difficulties, the research was halted in favor of the more productive Weak AI research. Instead of having a general intelligence capable of solving any type of problem, the research was directed to more specialized intelligences, each one capable of solving a specific set of problems.

With this approach, although resulting in smaller projects and results, it was possible to increase the development speed due to the division of research in simpler branches of AI and the number of these branches. Over the times there have been much advancement in AI, and today there are several products based on AI research incorporated as an every day's item, without the consumer even knowing the origins of the product. Some examples are the optical character recognition (OCR) [4] that comes with almost any image scanner, the AI opponent on any computer game available, or even the Kinect 3D-body-motion interface for the Xbox 360 gaming system [5].

While games have taken great benefits from AI research – especially for non-human player characters – the field also has taken great interest in games. This interest comes from the well-defined and easy to understand rules of the game, and the problem of defining the best strategy for winning. One such game is chess, the rules of the game are simple to understand: each player can only make one movement per turn; each piece has its own type of movement; the game ends when one of the players has the opponent king in a checkmate. Despite these simple rules, the strategy needed for winning is very complex, normally neding to plan moves several turns ahed of the current one. The most notable result in chess AI research was the victory in 1997 of IBM's super-computer Deep Blue over Gary Kasparov, the world chess champion at the time [6]. Recently, IBM also gave a big step in another game/contest, Jeopardy. The game's objective is to discover the question to a series of answers, and the theme varies from history, arts, science and literature, among others. In this case, IBM's computer, Watson, managed to win against two former champions by a large margin [7].

Since Deep Blue, Poker has been the most interesting game for AI research because it presents challenges not found on chess or other similar games. Of these challenges, there are two important, the presence of randomness and the absence of information. The first, the random events, results from the shuffle of the cards in the beginning of the game, it is impossible to have an exact knowledge of what cards will be dealt to whom and when. The second challenge, the incomplete information, is due to the secrecy of each player cards, this is, each player only knows its own cards. Just these two facts take the results of chess research one step forward, as it is a better representation of a dynamic environment.

1.2 Motivation

There are two main reasons that motivate poker research.

The first one is Poker itself, as part of the entertainment industry it has been growing on popularity for several years, and while its popularity was already high in the casinos, it had an even higher growth in online casinos [8]. This growth combined with the easier access to the web, made poker a very popular game to the point of receiving media coverage as a conventional sport. A good example of this is the Poker Channel [9], a TV channel exclusively dedicated to Poker competitions, news, events, etc.

The second reason that motivates poker research is the challenges that it presents. While the research with chess had a big success and was a big step for AI, for the outside world it didn't seem to have the same usefulness. The applications in the real world that could benefit from the research with chess are limited, as it is unusual to have complete information about anything, and even the information available may change with time and/or location. This is one of the challenges that the real world presents, and one that is tackled in the research of Poker.

Aside from this, there are other challenges like random events, risk management, bluffing and multiple opponents. All of these are challenges present in the real world and that are also a part of the game of Poker. All these aspects of the game turn Poker into an interesting research domain, with several challenges and future practical applications.

Aside from computer science, Poker has also aroused interest in other fields such as economy [10] and psychology [11].

1.3 Goals

As with all computer Poker research, the general goal of this thesis work is to contribute to the poker research community by developing an intelligent agent capable of playing Poker.

The agent developed in this work will be using a new approach, Reinforcement Learning. The objective of this approach is to test whether it is possible for an agent to be able to learn how to play, and, at the same time create strategies that could give it the upper hand against different type of opponents.

Another goal of this project is to ascertain the best way of describing the state of the environment for the game of Poker. Althought not being a vital objective, this can make a diference in both performance and final results, since it can not be either too generic or too strict.

1.4 Summary of Contents

This document is divided into six chapters.

In the first chapter, this one, a small introduction of the problem is made, also presenting the motivation, context and goals for the work. The document's structure is also described here.

The second chapter describes the game of poker, its rules, its importance to AI research and also a small description of the No Limit Texas Hold'em game variant.

In the third chapter is presented the state of the art of Computer Poker research, tools and other projects related to this own work.

The fourth chapter presents the approach taken to solve this thesis problem, first by explaining the logic behind the solution, and then the implementation design of the agents.

The fifth chapter presents the results obtained after the experiments of each developed agent.

Finally, in the sixth chapter, a small explanation of the results is presented, as well as a conclusion taken from this work and possible future work in this area of research.

Chapter 2

2.Poker

2.1 Poker Overview

2.1.1 What is Poker

Poker is a generic name attributed to several card games where players bet that the card combination in their possession have a higher value than those of the other players. The winner of the game is the one who has the card combination with the higher value or the player who remains in the game if all its opponents folded.

2.1.2 Poker Origins

The origin of poker has been, and still is, a well-debated topic, with as many variations regarding the birth place of poker as the variants of the game itself. One theory places the time and place as 16th century Persia in a game called Âs Nas [12], it was similar to the current 5 Card Stud variant of poker and it included betting rounds and the use of hierarchical hand rankings. While this is a popular theory, it is often refuted due to the absence of any description of the game before 1890, and because the word "Âs" is not a card related word in Persian, it most likely derives from the French word for ace [13].

These arguments lead several researchers to believe that the Âs Nas game was inspired by a European vying game. There were several games at the time that shared parts of the unique combination of mechanisms found in Poker, although in different forms. One of these games was from Germany, called Pochen, and its French variant, the Poque [14]. Both these games contained some of the aspects of the current game of Poker, such as hand ranking, betting and bluffing.

2.1.3 Poker Growth

While the origin of the Poker can be theorized to be European, it was not in the same form as the current game. The earliest reference to Poker occurs in 1836 [15], which time the birth of contemporary Poker at the 19th century. It appeared in a former French territory around New Orleans [16], which became part of the United States of America by the year 1803.

The first known form of Poker used a 20-card deck, containing Ace-King-Queen-Jack-10 cards dealt evenly amongst four players. Due to the limited deck, the bets were made on a narrow range of hand combinations: one pair, two pair, triplet, full and four of a kind. Unlike what happens in current variations of Poker where the best hand can be tied with another suit, in this form of Poker, the top hand of four Aces or four Kings and an Ace were absolutely unbeatable [17]. Through time, the game started to adopt a 52-card deck, which allowed more than four players, it also allowed the addition of a new hand ranking: the flush, and the most important change, it provided enough cards for the introduction of Draw [18].

The civil war that swept the United States of America in the 19th century saw Poker experience many more changes and innovations, leading to structural divisions of the game, giving birth to new variants. One of the most notorious was the Stud Poker, credited to be a cowboy invention by the year 1864 [19].After this, around 1920, there was another structural division of poker, where one or more cards are shared between all players, these new games were denominated as Community Poker, being Texas Hold'em variant the most popular.

2.1.4 Betting Structure

Betting is the key to win in a Poker game, since it allows to minimize the losses when holding a bad hand and to maximize the wins with good hands. Betting is typically done in clockwise order. When it reaches the player's turn, the available actions are:

- **Check**: This option is only permitted if no player has already bet in the current round. By checking, the player retains the right to call or raise any bet made subsequently by another player. A check is considered a bet of zero.
- **Call**: To call a bet is to wager enough to match what has been bet into the pot since the last time any player bet.
- **Bet**: A bet is a wager of a certain amount of chips or money. The amount of a bet may be limited by the rules of the game.
- **Raise**: The raise is a particular kind of bet. To perform a raise, the player first has to bet enough to match what has been bet since the last time it was his time to bet and then increases the amount wagered by a new amount. This new amount may

be limited or not, depending on the type and rules of the game. Although this choice seems like a two-step operation, it works as one.

- **Fold**: If a player decides not to choose any of the above actions, then the player can fold. This drops out the player's hand, relinquishing any possibility of winning the pot.
- **All-In**: A special case of raise where the player bets all of its remaining chips or money.

Betting continues until everyone call or folds after a raise or initial bet. At the end of the round, the highest hand still in the game wins the pot.

2.1.5 Poker Variants

Aside from the Texas Hold'em variant of poker used for this work, there are several others variants played. Some of them are described in this section.

2.1.5.1 5-Card Draw

This type of Poker rose from relative obscurity during the American Civil War to become the most popular game for almost a century.

As with some other games, an ante must be paid, with the amount varying for each game, just to get the cards dealt. After paying the ante, each player receives five cards face down. Then, starting from the player to the left of the dealer, each player chooses its betting action. Once the first round of betting is over, each active player has the option of discarding from one to five cards, if the rules of the game don't restrict it, and receive replacements from the dealer. After the draw, there is a final betting round, usually starting with the player who opened the pot. In the showdown, the player with the best high hand wins. This game allows the use of jokers as wild cards [21].

2.1.5.2 Straight Draw

Straight Draw Poker is in all similar to 5-Card Draw but there are no wild cards. There can still be restrictions to how many cards a player is allowed to draw each time.

2.1.5.3 7-Card Stud

Shortly before the Second World War, this type of Poker became the most popular variant and maintained its position for about 40 years, mostly with the help of the new and thriving Las Vegas casino industry after the state of Nevada legalized casino gambling.

The game starts with the dealer giving two cards face down to each player and one card face up to each player. The player with the highest card showing opens the first betting round. Following this betting round another card is dealt face up to each player, followed by a betting round, followed by a third card face up, followed by a betting round, followed by a fourth card face up, followed by another betting round, followed by the last card face down, concluded by the final betting round.

The player that opens each betting round is the player that has the best hand showing out of the cards face up. In the end, each player takes five cards out of the seven that make up the best hand, with the best high hand winning.

2.1.5.4 Razz

Razz is played like 7-Card Stud. The twist is that in Razz, the lowest hand wins. Each player is dealt two hole cards, meaning they're dealt face down, and one card faced up. The dealer then gives each active player three more cards facing up, and then a final card facing down. Each player ends up with seven cards, four faced up and three faced down. At the showdown the player holding the best low hand using only five of his seven cards wins the pot. In this game aces are always low and flushes and straights have no effect on the value of a hand.

2.1.5.5 Lowball Draw

In Lowball the lowest hand at the table wins the pot. As with Razz, straights and flushes are ignored in Lowball, although some tables count them as high, and therefore contribute to a bad Lowball hand.

The dealer starts by giving each player five cards face down. There is a round of betting, starting by the player to the dealer's left. After the initial betting round, players may draw out up to five cards. Following the draw there is a final round of betting. Usually the rules of play require a 7 low or better to bet in order to win any money to put in the pot after the draw. The lowest ranking hand in the showdown wins the pot. Frequently the joker is used as a wild card [22].

2.1.5.6 High-Low Split

This name covers several popular forms of Poker. Essentially in High/Low games the pot is split between the best hand and the low hand at showdown. This is a feature that can be added to just about any Stud Poker game, so the game can be 5-Card Draw, 5-Card Stud or 7-Card Stud, in addition to other game's rules. Sometimes, however, the rules may require that players declare whether they are going for the high, for the low or both. Like in Lowball Draw, in 5-Card High/Low split games, the best low hand is always A-2-3-4-5. Omaha/8, a variation of

High/Low split, requires a player to have 8 low or better to qualify for low. If no one has an 8 low or better, the best high hand wins the whole pot.

2.2 Texas Hold'em

At the moment, the Texas Hold'em variant of Poker is the most popular game of its kind, and it is also the variant considered for this work.

After its introduction in 1920, the Texas Hold'em game had a slow rise in popularity until 1998, were it had a boost with the release of the movie "Rounders" [20]. The movie tells the story of a poker player and its adventures in the underground poker world. Due to the dramatic representation of the swing that a poker player faces in this game, it became very appealing to people all around the world. Over the next two years, due to the sudden interest in the game, a lot of literacy had come to the stores, available to anyone.

2.3 Rules

Texas Hold'em is a game that uses the player's position at the table to strategically improve its game. This is achieved through the use of buttons, at any given moment there is a dealer button attributed to a player, this mark the top position at the current table and also determines who the small and big blind are. These are the next two players sitting to the left of the dealer. These positions are rotated clockwise at the end of every round.

The Figure 2.1 shows the dealer as the player at the position F, followed by the players in the positions A and B as small and big blind respectively.

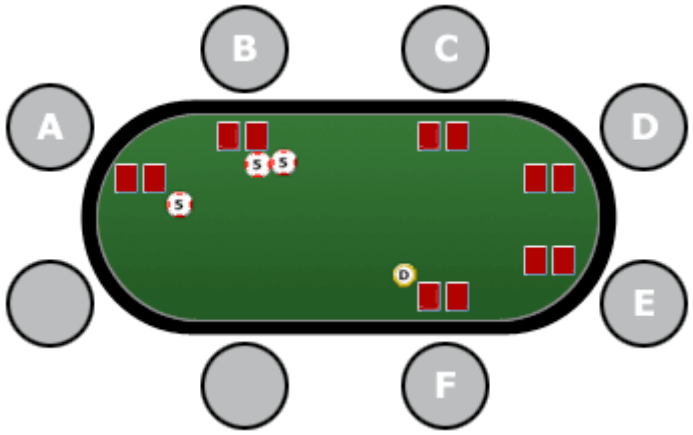

Figure 2.1 Texas Hold'em Table during Pre-Flop

The first round of the game is called Pre-Flop. At the start of the round, each player is dealt two hole cards, and then both the small and big blinds bet a predetermined amount of money to the pot, the small blind is half the minimum bet and the big blind is the minimum bet for the table. This is to ensure that there is action in every hand. Then the remaining players choose their actions for the betting round until everyone calls or folds.

The second betting round is called Flop. Three community cards are dealt face up in the middle of the table. These cards are shared by all the players and can be used in combination with the hole cards that each player holds. After this, the second round of bidding begins, and only ends when all the players put the same amount on the pot or fold.

After the end of the Flop round, the dealer turns one more community card face up in the table and a new betting round begins. This third round is called the Turn, and proceeds similarly to the previous one.

The fourth and last round is the River. The last community card is dealt and displayed in the table, starting the last betting round. This can be seen in Figure 2.2.

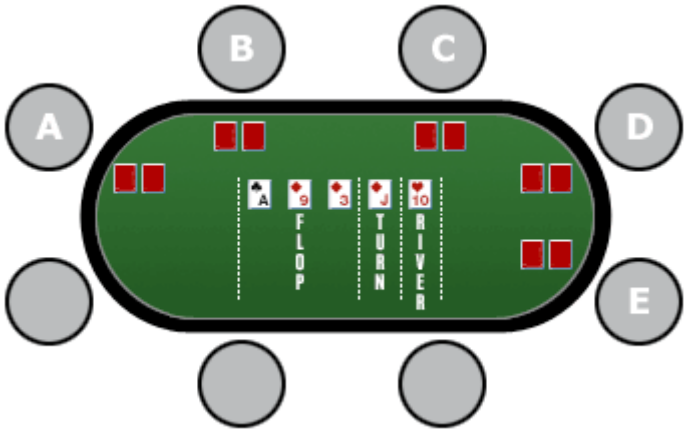

Figure 2.2 Texas Hold'em Table during River

In any of the previous round, if all but one player folds, then the remaining player is declared the winner of this hand and wins the pot on the table. At this point, the player may choose to show his hands to the other players, or muck it, which means to throw the hand away without showing anyone what it was.

On the other hand, if after the River, there are still two or more players in the hand, a Showdown phase takes place. In this phase, all players show their cards, starting with the last person to bet. However, after the first player shows his cards, the other players may choose to muck their hand, which is basically the same as folding. This is an important part of Poker as you can muck to keep other players from learning your playing style. The players can use any combination of seven cards to form the best five-card Poker hand.

2.4 Hand Ranking

In Poker, certain card combinations, or hands, outrank other hands. The player with the best hand at the Showdown wins the pot.

The following general rules apply to evaluating poker hands, whatever set of hands are used:

- Individual cards are ranked A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2, A. Aces only appear low when part of an A-2-3-4-5 straight or straight flush. Individual card ranks are used to compare hands that contain no combinations.
- Suits have no value. The suits of the cards are mainly used in determining whether a hand fits a certain combination (specifically the flush or straight flush hands). In most variants, if two players have hands that are identical except for the suit, then they are tied and split the pot.
- A hand always consists of five cards. In games where more than five cards are available to each player, the best five card combination is the hand of those cards players.
- Hands are ranked first by combination, then by individual card rank: even the lowest qualifying hand in a certain combination defeats all hands in all lower combinations. The smallest two pair hand, for example, defeats all hands with just one pair or high card. Only between two hands in the same category are card ranks used to break ties.

The deck used in Texas Hold'em is a 52 card deck as shown in Figure 2.3.

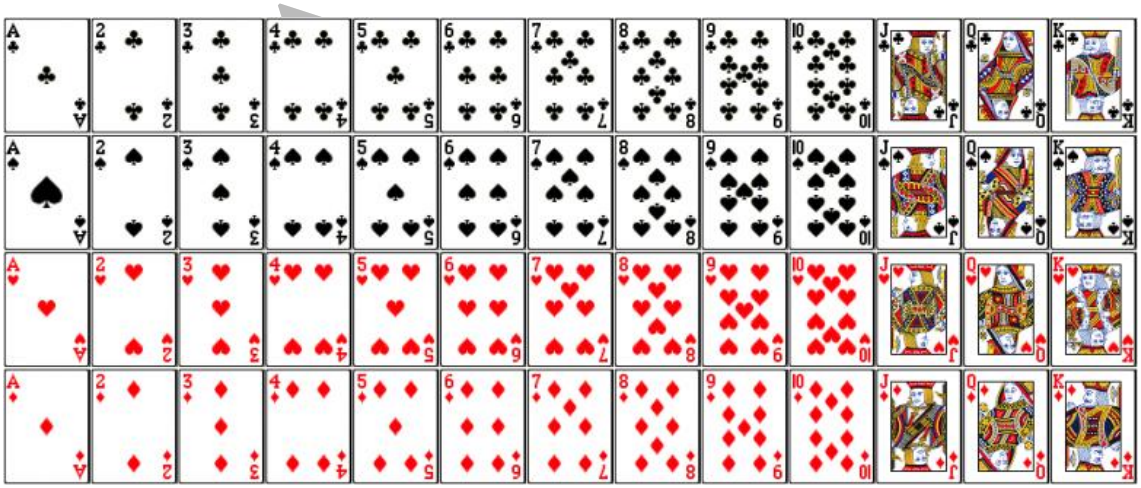

Figure 2.3 52-card deck

The combinations of cards are described next in order from the highest to lowest.

2.4.1 Royal Flush

Figure 2.4 Example of a Royal Flush hand

The Royal Flush is the highest ranked hand available in Poker without wild cards. This is a particular sequence of the Straight Flush hand with the highest ranked cards.

2.4.2 Straight Flush

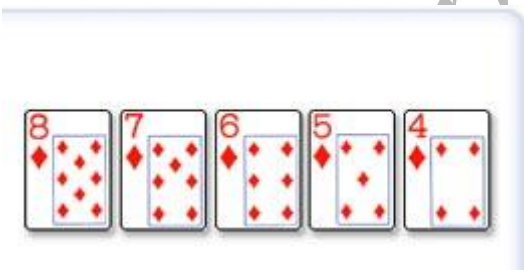

Figure 2.5 Example of a Straight Flush hand

The Straight Flush is a hand containing five cards in sequence, all of the same suit. When two players go to showdown with this hand, the hands are compared by the highest ranked card. Since the suits don't have a relative value, two identical Straight Flush hands with different suits tie. Aces can play high or low in Straights and Straight Flushes: A-2-3-4-5 is considered a 5-high Straight. The Royal Flush mentioned above is a particular case of the Straight Flush.

2.4.3 Four of a Kind

Figure 2.6 Example of a Four of a Kind hand

Four of a Kind, also known as quads, is a hand where the player has four cards of a certain rank and an unmatched card of another rank. In Texas Hold'em, due to the community cards, it is possible for two or more players to have the same quad. In these cases, the kicker – the unmatched card – is used as a tie breaker.

2.4.4 Full House

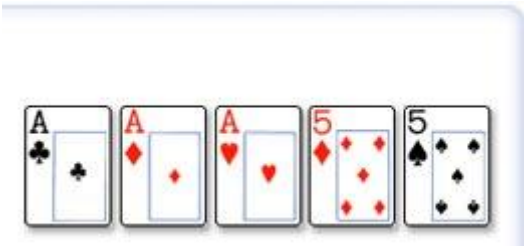

Figure 2.7 Example of a Full House hand

The Full House is a hand where all the cards are active, this is, contains a set of three matching cards of the same rank, and two matching cards of another rank. Between two full House hands, the one with the highest ranking set of three wins. Again, due to the community cards of Texas Hold'em, there is a chance of two or more players to have the same ranking in the set of three, so, in these cases the hand with the highest ranking pair wins.

2.4.5 Flush

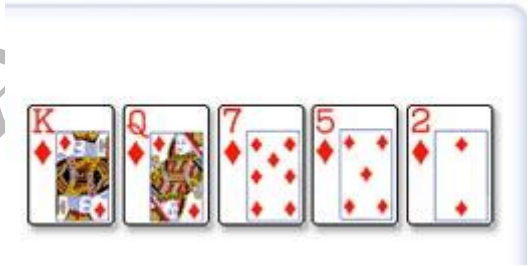

Figure 2.8 Example of a Flush hand

The Flush is a hand containing five non sequential cards of the same suit. If two players have this hand, the one with the highest card wins. If both have the same higher card, then it compares the second higher, and so on until a difference is found. Since the suits are not used to rank the cards, if both players have the same cards, with different suits, they are tied.

2.4.6 Straight

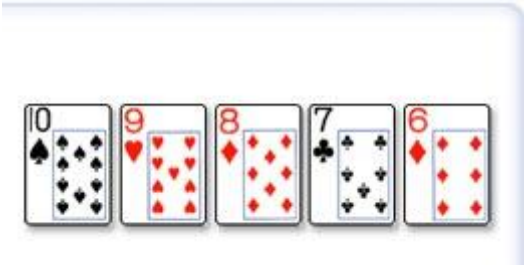

Figure 2.9 Example of a Straight hand

A Straight is a hand containing 5 sequential card ranks, but with different suits. In case of two Straight hands, the one with the higher ranked card wins. If the higher ranked cards of both hands are of the same rank, the hands are tied since the suit doesn't differentiate them.

2.4.7 Three of a Kind

Figure 2.10 Example of a Three of a Kid hand

Three of a Kind, also called trips, is a poker hand containing three cards of the same rank and two unmatched cards. A Three of a Kind hand wins another if the set has a higher ranked card. In case of both sets containing the same ranks, the higher ranked of the unmatched cards are compared to break the tie.

2.4.8 Two Pair

Figure 2.11 Example of a Two Pair hand

A Two Pair is a hand containing two cards of the same rank, another pair of matched rank, different from the first, plus an unmatched card. To rank two hands of this type, the higher ranked pair of each is compared, and the higher one wins. In case both players have the same higher ranked pair, the second pair is compared. Finally, if both hands have the same two pairs, the kickers determine the winner. There is also a possibility of a tie if both hands have the same ranked cards.

2.4.9 One Pair

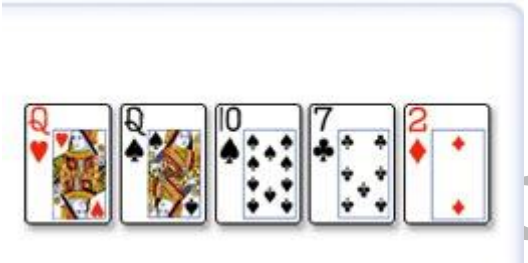

Figure 2.12 Example of a One Pair hand

One Pair is a hand containing two cards of the same ranks and three other unmatched cards. In a Showdown with two hands containing One Pair, the one with the higher ranked pair wins. If the pair is of the same rank in both hands, the kickers are compared in descending order until it is possible to determine the winner. If both pairs and kickers are of the same rank in both hands, the game is tied.

2.4.10 High Card

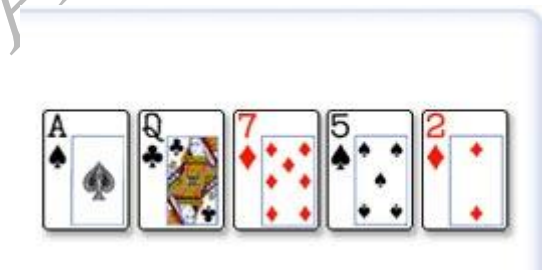

Figure 2.13 Example of a High Card hand

High Card hand is a combination in which no two cards have the same rank, the five cards are no in sequence and all cards are not of the same suit. Two High Card hands are ranked comparing the highest ranking card. If those are equal then the second highest ranking card is compared, and so on until a difference is found. If both hands are similar in rank, the game is tied.

2.5 Importance to Artificial Intelligence

Normally, games can be classified by two parameters:

- Available information;
- Whether the element of chance is accepted or not.

The first parameter represents the information that is available to the player. A game with complete information is a game that anyone, at any time can identify the whole state of the game. On the other hand, in a game with incomplete information, only partial information is available to each player, which means that it's not always possible to correctly predict the outcome of an action.

The second parameter represents the involvement of chance in the game. A deterministic game doesn't allow the element of chance, which means that the next state of the game is purely defined by the current player's actions, without any external factor. A nondeterministic game allows external factors to influence the state of the game, one example are random events. A nondeterministic game can also be called a stochastic game.

Figure 2.14 shows some example of games and their classification with these two parameters.

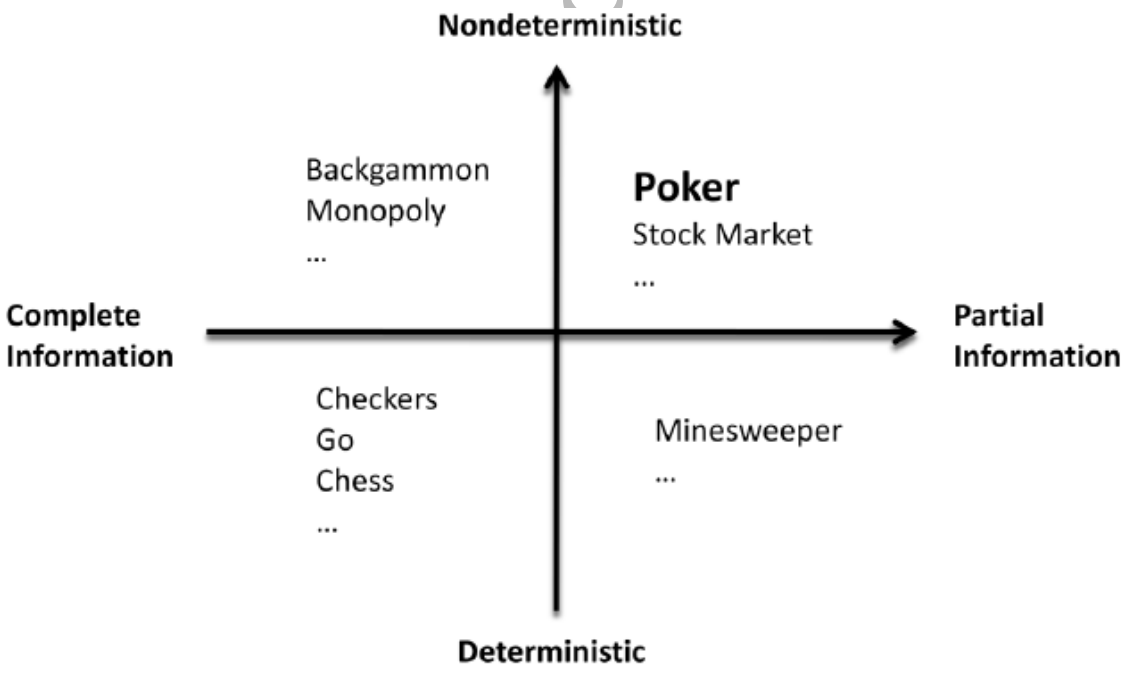

Figure 2.14 Game classification

As it is possible to observe, Poker is a stochastic game with partial information. These two characteristics of Poker represent several problems to solve in the field of AI.

From a game-theoretic perspective, it is unfeasible to compute an optimal solution to the problem in the poker's domain. Even the two player Texas Hold'em game has a search space of O (10^{18}) [23].

The outcome of every action depends immensely on the actions of the other players, and of the accuracy of the model constructed for them. Even an optimal poker strategy (one that minimizes the loss against any possible opponent's strategy) could prove not to be as effective as another one (maximal strategy), which would better exploit the opponent's weakness [23]. This is the reason Opponent Modeling is a main component of any good Poker program.

Furthermore, humans make use of their intuition in Poker, so they change their playing style very fast to adapt to their opponents style and to avoid being accurately modeled. This issue was discussed in the past, as "tracking and moving targets" [24].

2.6 Conclusions

In this chapter the game of Poker was described giving more attention to the Texas Hold'em variant.

It was possible to see the simplicity of the game and its rules, but also the complexity of the strategies needed to have a good performance against other players.

Chapter 3

3.State of the Art

3.1 Building a Poker Playing Bot

There are mainly three different types of approaches for building artificial poker players: Heuristic-based, Simulation-based and Game Theoretic-based. These approaches provide a specification, a strategy or a policy on how a player should react in face of each situation that can arise during the game.

The next subsections present some approaches for strategy development.

3.1.1 Rule Based

The first intuition regarding the development of a computer program capable of playing poker is to define a series of conditional if-then-else rules, specifying the actions that should be taken for each of a large set of situations that can arise. This approach is similar to the way human players describe how they play, on a case-by-case basis. For this reason this is also known as a heuristic-based approach.

Although intuitively reasonable, this approach has been proven to be extremely limited [25] largely because, in theory, it remains overly simplistic, since the abstraction of trillions of possible situations onto a much smaller number of general circumstances is unable to capture the subtlety and nuances of a strategically complex game like Poker.

3.1.2 Formula Based

Another example of a heuristic-based approach is the formula based approach. This one typically resorts to an arbitrary complex procedure or formula to use as a criterion for distinguish cases. For example, it can have a weighted enumeration of sub-cases used to

determine a hand's value and consequent probabilistic action based on predetermined thresholds for that value. This can effectively multiply the number of distinct situations able to be identified, creating a more flexible generalization than what is achieved with those of rule based [26].

Despite being more flexible, formula based approaches still rely on the same principle used by their rule based counterpart. As a result, although to a lesser extent, much of the rule based liabilities are also present in the formula based systems. These are also complicated to create and maintain as more situations are abstracted onto the system.

3.1.3 Simulation Based

Simulation based approaches consist in studying the repetition of many instances in order to obtain a statistical average. One technique proven to be powerful is the Monte Carlo simulation, relying on repeated computation and random or pseudo-random numbers, for modeling phenomena with significant uncertainty in inputs like the calculation of risk in business. With purely random samples, it can take a long time for the simulation to converge on accurate estimates. In order to accelerate the process, a certain degree of biased sampling is usually introduced, like selective sampling which focus on samples that provide the most information gain [27].

Unfortunately, in practice, this approach can be highly volatile and result in extremely unbalanced play, since the quality of the simulations depends on the quality of the simulated play. This makes it vulnerable to too much biased values for certain situations, leading to inaccurate plays [28]. Also, in poker, future actions must not be determined by depending on explicit knowledge of opponent's cards in each simulation since observation of perfect information instances cannot, in general, produce accurate results for an imperfect information situation [29].

3.1.4 Nash Equilibrium

Game Theory is a branch of mathematics and economics that is devoted to the analysis of games. This is especially relevant since games can be used as models to many real-world decision problems. Nash Equilibrium is a concept developed from Game Theory.

A Nash Equilibrium is a strategy, for each player, with the property that no single player can do better by changing to a different strategy, meaning that no player has an incentive to deviate from the purposed strategy because the alternatives could possibly lead to a worse result [30]. This implicitly assumes the opposition of perfect players, players that will always do the best possible move, which in real poker, is definitely not the case since players are highly

fallible. Nevertheless, a Nash Equilibrium strategy represents a great achievement, especially in two-player zero-sum game, like in heads-up poker. If both players are based on this approach, the expected score for both of them will be zero. On the other hand, if only one player has a Nash Equilibrium approach, he can expect to do no less than to tie the game, since the opponent cannot do better by playing a strategy other than the equilibrium. A thorough description of this game theory solution can be found in Oliehoek Master's thesis [31].

The Nash Equilibrium problem for two-player zero-sum sequential games can be modeled using linear programming. Unfortunately, creating a perfect Nash equilibrium strategy for a complex game like Texas Hold'em is extremely difficult and, at the moment, computationally unfeasible. Instead what is currently used is ε-Nash Equilibrium, an approximation to the Nash equilibrium strategy, resulting in a suboptimal strategy that proposes a best response instead of the perfect response. ε is the value of the best response to the determined suboptimal strategy and is a measure of how far from the actual equilibrium the strategy is. If an opponent, either human or machine, makes mistakes then the equilibrium strategy can win over time.

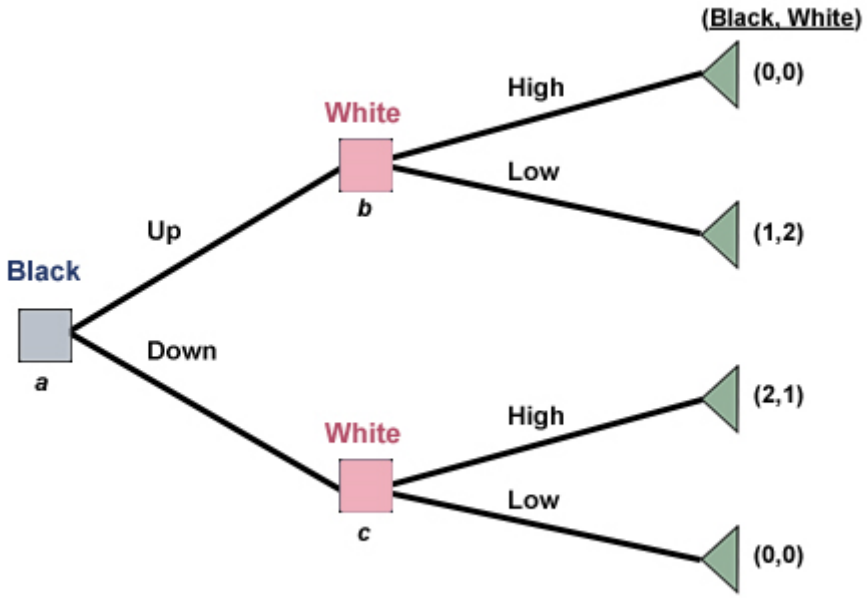

Figure 3.1 Example of an equilibrium sub-game: White plays low if Black plays up, and plays high if Black plays down

The success of this approach can be measured by the number of artificial poker players created with this strategy as a core. CPRG has developed a wide range of these players namely the PsOpti series and the Smallbot series. Researchers from Carnegie Mellon University also created a series of artificial players based on ε-Nash equilibrium strategy [32]. Notably, even non-institutions have created these players as is the case with Bluffbot, a creation of Teppo Salonen [33].

Despite the success [34] of this approach, there are implicit problems with it [35]. Adding to the already mentioned accuracy tradeoff and the liability against non-optimal players, a Nash equilibrium strategy is a fixed strategy, meaning that despite being more or less time consuming, once a flaw in the strategy is discovered it can be repeatedly exploited. Also, most of these computer poker programs are oblivious to an opponent's strategy and can easily become prey to probing for weaknesses without fear of being punished for using a highly predictable style.

In short, artificial poker players based on ε-Nash equilibrium strategy play close to the optimal strategy, making it near-unbeatable by any other strategy in that abstraction. This is particularly useful since it allows defense against optimal/near-optimal opponent strategies and/or safely learning of an opponent tendencies for several hands before attempting to exploit them. Also there are situations where obtaining a quick best response can compensate for the expected cost of computing the perfect response.

3.1.5 Best Response

The best response approach is based on the paradigm that for every strategy, there is a best possible response. Calculating the best response to the opponent‘s strategy is very computationally expensive, so an approximation to the best response strategy is usually the solution. This approximation is called abstract game best response. The way this approach works is by choosing the action with the highest value of utility, at every information set, from the probability of the current strategy reaching every terminal node from every possible game state from that point on and the utility value associated with each terminal node. A formal description of this algorithm is presented in Jonathan Master’s thesis, chapter 4 [30].

In practical terms what this means is that this approach when facing ε-Nash Equilibrium strategies, for example, is able to determine the value of ε and therefore capable of determining the lower bound on the exploitability of that particular strategy. This is important because poker is a game where exploitation of opponent‘s weaknesses is crucial. The objective in Poker is not to not lose against an opponent but rather making sure one wins the most against an opponent.

Although promising, the abstract game best response approach has requirements that limit its use in poker games like Texas Hold‘em. First, the algorithm requires knowledge of how the strategy acts at every information set; so unless the opponents play in a predictable way or chooses to provide details regarding their own strategy it is difficult to calculate an abstract game best response to an opponent‘s strategy. Second, the abstract game best response has to be calculated in the same abstraction as the original strategy.

In short, the best response strategy computes the maximizing counter-strategy to a given static strategy. A match between a program and its abstract game best response allows the latter

to determine by how much the program can be beaten. Although being a useful tool for evaluating other strategies, by itself its usefulness is limited against arbitrary opponents due to its requirements.

3.1.6 Adaptive Programs

As it should be coming clear by now, a stationary poker playing strategy is easily vulnerable to be exploited. To be able to play poker at a world class level, a player must be able to assess the game‘s conditions and adjust to any special circumstances. This ability is essential to mastering the game of poker, whether the player is a human or a program.

The adaptive modeling system has two properties at its core: accuracy is limited by the correctness of the relative weights assigned to each future action of the opponent; and by the equity estimations for each showdown outcome. What this translates to is that for every combination of future chance outcomes, the net loss or gain of every future betting sequence is considered, with each being weighted by its relative likelihood. The decisions are made by using an adaptive imperfect information game-tree search, specifically the Miximax and Miximix algorithms [27]. The adaptive approach mimics the type of calculations done by human poker players, albeit at a much higher level of precision and accuracy.

In determining the correct mathematical play to make, adaptive programs can be seen as simply computing a best response but on current opponent‘s beliefs, which are subject to change over time. In principle this is correct but these advanced systems also refrain from continuously using the best response available, deviating from a simple best response for added benefits. There are several reasons for this:

- avoid predictability, since pure best response without modifications represents highly predictable reactions to certain situations;
- avoid exploitation, since against a player who is constantly changing styles, the over-reactive best response opponent may swing like a pendulum between belief states;
- increase unpredictability, since patterns are harder to be identified by opponents;
- and the pursuit of exploitation, because it is more profitable to exploit an error at a slower sustainable rate, so that the known weakness persist indefinitely, than it is to punish an opponent‘s error too severely and lead them to change their behavior [36].

Although this approach has numerous advantages over the previous approaches, it also has some practical limitations:

First, it is mandatory to have good default data since the system allows for essentially any belief to be held, regardless of how irrational it might be.

Second, the construction of knowledge is done by observation (of opponents and hands) and this technique requires a considerable amount of data to be effective.

Third, data scarcity can also be a problem in some designs since the structure of the imperfect information game tree provides a natural separation of contexts which must be combined according to greatest similarity.

In short, the adaptive programs approach provides a very interesting set of properties that fulfill the requirements in order to build a world class poker player as defined by Billings et al. [37].

3.1.7 Restricted Nash Response

The Restricted Nash Response [30] is a strategy that combines the Nash Equilibrium and the Best Response approaches fairy well. It was designed to solve the tendency to lose to arbitrary opponents found in the best response strategies.

The Restricted Nash Response essentially creates a strategy that is designed to exploit opponents but will do so in a robust way, which is a way to exploit a particular opponent or class of opponents, while minimizing its own exploitability. In practical terms this means that if the strategy being played was trained against the opponent, victory is assured; if the strategy being played was not trained against the opponent then in case of defeat, it will not be by much.

This approach initially uses the Frequentist Best Response algorithm to create a model of the opponent‘s play and then resorts to the Counterfactual Regret Minimization algorithm to find an ε-Nash Equilibrium. This will generate counter-strategies that provide different tradeoffs between exploitation and exploitability. The generated counter-strategies are in the set of ε-safe best responses for the counter-strategy‘s value of ε, making them the best possible counter-strategies, assuming the model is correct. ε-safe refers to a strategy that cannot be exploited more than ε. Crucial to this approach is also the strategy‘s parameter $p \in [0, 1]$ that represents the degree of confidence or belief in the accuracy of the model. The higher the p, the more it moves away from the Nash equilibrium.

There are several advantages with this approach. The Restricted Nash Response counter-strategies are robust responses to opponents, unlike traditional best responses which tend to lose against arbitrary opponents. This robustness makes the Restricted Nash Response a good candidate for use against an opponent who is suspect of exhibiting an exploitable trait. This strategy is also more efficient, capable of computing the same information as the Best Response

but in a smaller amount of time; and more effective, being capable of achieving nearly the same exploitative power as best responses, but with only a fraction of the exploitability.

There are also a couple of limitations with this approach. If the generated model is inaccurate or incomplete due to a limited number of observations of the opponent‘s actions, the restricted Nash response strategy will perform poorly. This approach is not resilient to the case of no observations being made since it is unlikely that the default policy would accurately reflect the opponent strategy. More importantly, this approach is highly sensitive to the choice of training opponent, requiring a very particular set of observations, including full information observations, in order to perform well [38].

In short, the Restricted Nash Response is an approach for generating a range of strategies that provide good tradeoffs between exploitation and exploitability. The success of this approach has already been proven despite its limitations.

3.1.8 Data Biased Response

This recent approach was designed to partly solve the limitations found in the restricted Nash response approach [38].

The premise is that the selection of only one parameter, p, is not enough to accurately represent the problem since the accuracy of the opponent model is not uniform across all of the reachable information sets, like in the cases of limited or no observations whatsoever.

Instead of choosing only one parameter to reflect the accuracy of the entire opponent model, this approach assigns one probability to each information set I and call this mapping P_{conf}. Whenever the restricted player reaches I, the player will be forced to play according to the model with probability $P_{\text{conf}}(I)$ and choose actions freely with probability $(1 - P_{\text{conf}}(I))$. $P_{\text{conf}}(I)$ is set as a function of the number of observations of the opponent acting in information set I. As the number of observations of the opponent acting in I increase, more confidence is given to the model‘s accuracy. Noteworthy is the fact that if $P_{\text{conf}}(I)$ is set to 0 for some information sets, meaning that no observations were made, then the opponent model is not used at all and the player is free to use any strategy. Also noteworthy is the inclusion of Bayesian decision functions.

In practical terms, comparing to the restricted Nash response, there are several improvements.

First, data biased response doesn‘t require a default strategy. Like it was mentioned if no observation were made, any other strategy can be played although ideally it should revert to a self-play ε-Nash equilibrium strategy.

Second, it embodies "quality-assurance" since it sets a minimum number of observations in order to express any confidence in the model's accuracy while implementing linear and curve confidence functions for a trustworthy assessment of the model's accuracy.

In short Data Biased Response is, currently, the ideal approach for generating counter-strategies, especially since it avoids the outlined shortcomings of the restrictive Nash response approach while providing better performance in the most favorable conditions for the existing approaches.

3.2 Computer Poker Basics

The fundamentals of poker strategy are determined by the probabilistic nature of the game and the psychological understanding of the opponents. For artificial poker players, this is no different.

Artificial poker players, also known as bots, are expected to easily compute "poker's math" accurately. But poker is a mathematical complex game. Specifically, Limit Texas Hold'em has a search space size of O (10_{18}) [25] and the No-Limit version is even more complex. In order to accurately determine the correct mathematical play in a reasonable timeframe, it becomes mandatory to reduce the complexity of the game. To that end, researchers have come up with different ways of creating abstractions that make the game simpler to analyze.

This section describes the basic mathematical foundations of poker and the basic abstraction techniques utilized in more advance approaches.

3.2.1 Betting Strategy

Betting strategy in Texas Hold'em poker is usually separated into pre-flop and post-flop. The reason for this is that strategy in the pre-flop is significantly different from the post-flop stages; because no board cards have been dealt (flop) strategy at pre-flop is simpler than later in the game.

For instance, there are C (52, 2) = 1326 unique possible hands prior to the flop. Since there are no cards on the board, the suit of the card is irrelevant and many of these hands become equivalent. Using this knowledge there are only 169 distinct hand types in pre-flop Hold'em, which is far less than the possible 1.070.190 distinct combinations of two cards in post-flop stages (1081 possible opponent hands in the flop multiplied by the 990 possible turn and river cards) and this number increases exponentially the more opponents are playing. An exhaustive analysis of all match-ups of a player against nine opponents, in a Texas Hold'em game, requires

evaluating each possible board for each distinct starting hand against each possible combination of hands held by nine opponents, which is more than 21 octillion (approximately 2,117 x 10^{28}).

For this reason, a popular approach to the pre-flop strategy is to apply an expert system, be it based on empirical knowledge [39] or in simulations [27].

After the flop, complex algorithms and abstractions are utilized to simplify and determine a hand‘s value at each stage. Hands are not only ranked against other possible hands the opponents may have but also a hand potential is estimated, aiming to determine the chance that a hand has to improve or deteriorate in the next stages of the game.

3.2.2 Bucketing

Bucketing is a common and successful technique for drastically simplify games [25]. The idea is to partition the possible cards held by a player and on the board into buckets (sometimes called groups or bins) with the intent of separating hands that share similar strategic properties into the same bucket. This is very close to how a human would reason about a given hand. For instance, it doesn‘t really matter if the hold cards are a King and 2 or a King and 3 since both hands would probably be perceived as “a king and a low card”. Similarly, one possible approach for bucketing is to divide hands into buckets based on their strength such that weak hands are grouped into low numbered buckets and strong hands are grouped into high numbered buckets.

Assuming an even distribution of hands into buckets, if more buckets are used then each bucket will contain fewer hands. Since all hands in a bucket are played according to the same action probabilities, this leads to fewer cases where a strategy is forced to consider suboptimal action probabilities for a particular hand [30].

In the bucket abstraction, a strategy is defined over bucket sequences and not over cards. The bucket sequence is the sequence of buckets in which the cards were placed into on each round. For example, following the proposed approach, if a player had a weak hand on the pre-flop but the flop cards made it a strong hand then the hand may have been in bucket 1 on the pre-flop and now be in bucket 5 at the flop stage. To find the probability for every transition either sampling – fast and inaccurate – or enumeration – slow and accurate – techniques can be used. With the probability to move from each bucket at a betting round to another bucket on the next betting round, the model is built and the pseudo-optimal strategy created.

However, this is a one dimensional solution to a multidimensional problem. A hand cannot be categorized completely by only one parameter, so there may be subtle differences between hands in the same bucket that would require different action probabilities. Like all other abstraction techniques this is a compromise solution between the abstracted and the real game.

Nevertheless it is a powerful method of abstraction and advanced bucketing techniques have been developed [40] and used successfully in several different artificial poker players.

3.2.3 Opponent Modeling

As stated in section 3.1.6 of this work, "*to be able to play poker at a world class level, a player must be able to assess the game's conditions and adjust*". Opponent modeling is frequently considered to be a cornerstone of any competitive poker agent, be it human or machine, and rightfully so as it has been corroborated by the success of artificial poker agents – see section 3.4 – as well as demonstrated with dedicated research [41] [42].

Limited observations, stochastic observations, imperfect information, and dynamic behavior are the main challenges presented to opponent modeling but although these challenges are not yet completely overcome, there are techniques to cope with them. Currently two popular approaches (machine learning models) are used to build an opponent's model in poker: decision-trees and artificial neural networks. Both are based on statistic observation and both require a substantial amount of good data in order to be accurate.

3.3 Poker Tools

3.3.1 Pokersource Poker-Eval Library

The Pokersource [43] poker-eval library is, probably, the most widely-used poker hand evaluator**.** It's very fast, highly optimized, thoroughly examined for over more than ten years of use and includes support for multiple poker variants like Texas Hold'em, Omaha, 7-Card Stud, Low and High-Low games, etc.

The poker-eval library is implemented in a highly optimized, heavily macro'd C programming language but also provide language mappings for popular high-level languages like .NET, Java and Python. Everything is expressed either as a sequence of bits on which various operations are performed, or as a lookup table from which pre-computed values are stored. Poker hands are represented as a sequence of 52 bits, one for each card in the deck. This abstraction is called a card mask or hand mask and it can be used to store N number of cards, where N is any number between 0 and 52.

Figure 3.2 maps each of the possible 52 cards represented by a single bit in a 64-bit integer. Note that marked by X are 12 bits that are unused.

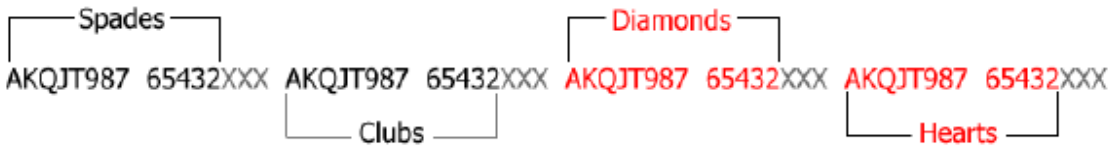

Figure 3.2 PokerSource card mask format

3.3.2 Poker Academy

Poker Academy is, probably, the world's most popular commercial software trainer for Poker.

Based on the research by the University of Alberta CPRG, it is a Texas Hold'em program that simulates the experience of playing in a real online poker room. It simulates both limit Hold'em and no-limit Hold'em, ring and tournament play. Its added value comes from the world renown AI used for its bots specifically Sparbot, Vexbot (for heads-up limit games) and Poki (for ring limit games).

For limit play other bots are available, for example Jagbot, a simple basic-strategy player with a complete inability to read opponents, or Simbot, which simulates the outcome of different possibilities in each stage in order to decide a move. For no-limit play, there is also Jambot, based on David Sklansky's "System" from [44] and Oddbot which deliberately makes random plays from time to time.

This software offers many other features but one is noteworthy, the Meerkat API [45]. Meerkat API is a Java API for developing bots in Poker Academy. This means that it is possible to create a custom computer opponent, plug it into Poker Academy and see how it does against both human and other AIs. For this reason, Poker Academy has already been used as a testbed for research on poker AI [46]. The Meerkat API available for download [47] contains the meerkat-api.jar file to compile, API documentation, instructions and a sample agent to start with. Unfortunately, knowledge of programming languages is extremely advisable (if not outright required) to fully use and understand this feature's capabilities.

3.3.3 ACPC Server

The ACPC Poker Server is an application made to simulate thousands of games between poker agents. This application is used to determine the winner of the Annual Computer Poker Competition [48].

The agents connect to the server and communicate with it using a TCP/IP protocol. The application is used for 3 different competitions:

- Two heads-up limit Texas Hold'em competitions
- Two heads-up no limit Texas Hold'em competitions
- Two 3-player ring limit Texas Hold'em competitions

3.3.4 Open Meerkat Poker Testbed

Open Meerkat Poker Testbed [49] is an open source implementation of the Meerkat API for running poker games. It imitates the Poker Academy simulator; however it is much faster because it lacks a heavy user interface.

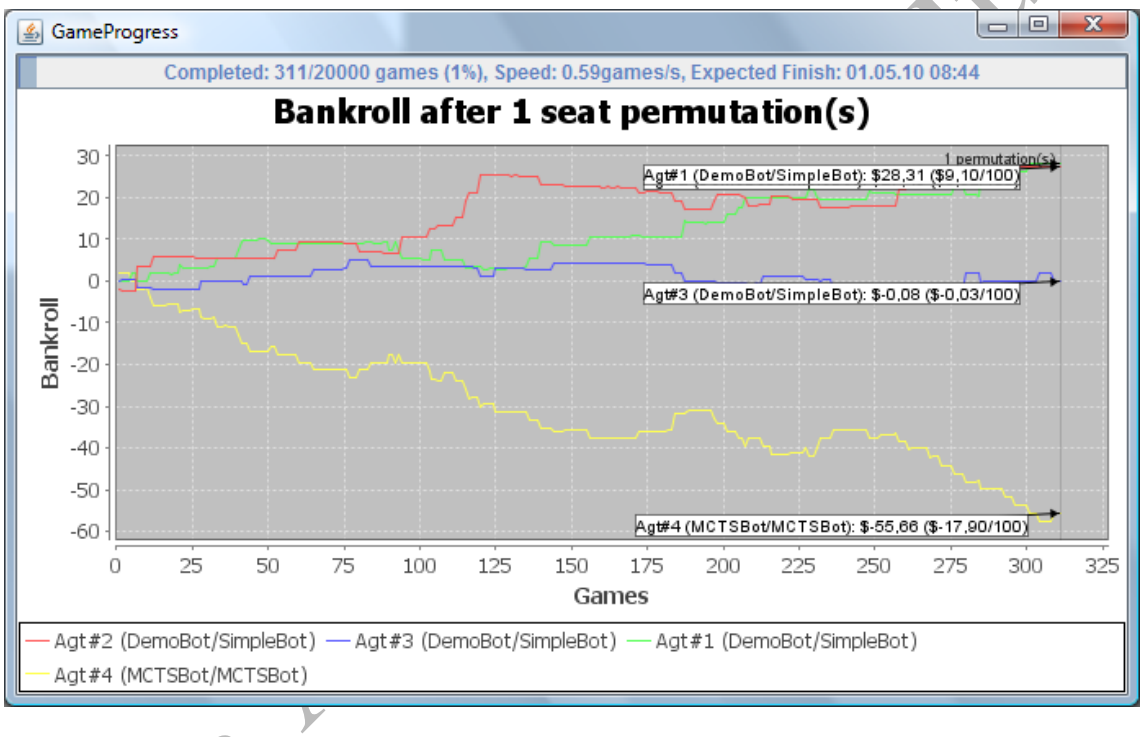

Figure 3.3 Open Meerkat Poker Testbed user interface

This application supports Fixed/No-Limit cash games with automatic rebuy. It generates bankroll evolution plots; implements seat permutation (replay games with same cards but with different seat order) and generate game logs. It also shows online bankroll evolution graph.

3.3.5 LIACC's Texas Hold'em Simulator

The Artificial Intelligence and Computer Science Laboratory (LIACC) is a research center at the University of Porto (UP), created in 1988. This center contains several researchers from the Faculty of Sciences (FCUP) and Faculty of Engineering (FEUP).

The center is involved in a large number of projects with national and international cooperation and it covers several subjects of investigation like Natural Languages Processing, Robotics and Distributed Artificial Intelligence among others.

Previously, this center had developed a Poker Simulator system called "LIACC's Texas Hold'em Simulator". The software was developed in C/C++ and it is based on a Server that communicates with several Clients through sockets under a predefined protocol. In this simulator, two versions of the Client were developed. One of the versions may be controlled by humans while the other is an autonomous agent that runs with an incorporated simples AI poker playing algorithm.

This AI algorithm is mostly based on the betting strategy described by Billings [25].

The implemented Server is capable to support ten different Clients (using a standard poker protocol based on TCP/IP) and allows the user to define all major characteristics of a poker game (initial stack, the value of the blinds). The number of simulations and the name of the log file that will save all hands played are also available to decide.

The protocol used by this simulator is compatible with the one developed at University of Alberta Computer Poker Research Group.

In Figure 3.4, we can see an example of LIACC's Texas Hold'em Simulator interface.

Figure 3.4 LIACC's Texas Hold'em Simulator user interface

3.4 Poker Agents

The number of autonomous artificial poker players, or poker agents for shorter, has been increasing in recent years. Many have been created during, and as a part of, academic research but as this research achieves more and becomes widespread, so does the number of individuals tackling and researching on this subject.

3.4.1 Loki

Loki was the first poker agent implementation made the University of Alberta CPRG. Loki [50] used a probabilistic formula-based approach, incorporating the GNU poker library [43] high-speed hand comparators as a core function, and expert systems designed by the author, to play (differently) each stage of the game. The play is controlled by two components: a hand evaluator and a betting strategy [48, pp. 3-6]. Loki was also developed using realistic games against human opposition. For this purpose the program participated in an on-line poker game [51], running on the Internet Relay Chat (IRC).

Although somewhat successful, Loki had limitations like requiring extensive knowledge, having complicated multi-component systems, being difficult to build on and having low accuracy for opponent modeling. Also, it was not capable of adaptive play, producing a single playing style.

3.4.2 Poki

Poki [52] is a complete object-oriented rewrite, also by the CPRG, of the previously mentioned Loki program. It is described as a simulation based system, which consists of playing out many likely scenarios, keeping track of the expected value of each betting action. Like Loki, it used expert system to guide the betting strategy but had neural networks introduced to improve its opponent modeling abilities and featured the miximax and miximix techniques to achieve more robust searches on imperfect game-trees.

Poki was also developed / tested resorting to IRC poker play, performing extremely well by consistently winning against human competition [27] of intermediate level playing strength. It performed so well that it became licensed for two widely popular commercial products, the video-game Stacked and software trainer Poker Academy.

Poki was designed to play full-ring (up to 10 players) limit Texas Hold'em but despite its success in ring games, in games with few opponents Poki's playing strength decreased significantly, becoming very weak in heads up games. This happened because the program couldn‘t adapt its strategy fast enough to exploit its opponents or prevent its own exploitation thus becoming too predictable and trusting of its opponents’ actions.

And so, it became clear early on that Poki's approach would be inadequate to achieve the goal of surpassing all human players.

3.4.3 PsOpti Family

This is the name of a series of artificial players designed to play heads-up limit Texas Hold'em. Created by the CPRG this series is known for its use of a game-theoretic approach (Nash equilibrium).

The PsOpti family of strategies is created by converting the abstract extensive game into a sequence form. The sequence form can then be transformed as a series of constraints in a linear program [53] and be solved to find the approximation to the Nash equilibrium. However, the linear programming required to solve the entire abstract game was considerable and additional simplifications like abstractions and/or separating the game into two phases were used. The techniques used to build this type of agents are described in detail in [34].

This series is currently up to its seventh version (PsOpti0 - PsOpti7). Their differences range from minor improvements over previous versions to play strategically different styles. For example PsOpti4 is less exploitable than PsOpti6 and PsOpti7, but PsOpti6 and PsOpti7 play a strategically different style that is useful against some opponents [30].

Noteworthy is the fact that PsOpti4 and PsOpti6 were combined to form Hyperborean06, the winner of the 2006 AAAI Computer Poker Competition [54]. Also noteworthy is the fact that PsOpti4 was licensed to commercial product Poker Academy under the name of SparBot.

3.4.4 Bluffbot

Bluffbot is a SparBot clone [25] created by an individual developer. According to its author, Bluffbot is a combination of an expert system and a game-theoretic pseudo-optimal strategy [33].

At its core is a handmade approximation of game theoretic equilibrium strategy based on the domain expertise of its creator. Its system includes various plays from professional poker players combined with pseudo-optimal bluffing, bluff catching and value betting strategies, which mean weighted decisions based on expected values. Human testing was also done to optimize the strategies even further [55].

The first version of Bluffbot competed in the first ever AAAI Computer Poker Competition [54] achieving the second place. The latest version, BluffBot 2.0 won the first place in the no-limit Texas Hold'em series of the second AAAI Computer Poker Competition in 2007 [56].

3.4.5 GS Family

The GS series are a Carnegie Mellon University creation. The initial GS1 poker agent [32] made use of a powerful abstraction technique to create an approximation to the GameShrink algorithm and build a competitive player at the time. Unfortunately the GameShrink approximation technique was discovered to have major drawbacks [57] and there was no statistically significant evidence to demonstrate that GS1 was better or worse than the competition.

GS2 [57] saw the introduction of an improved abstraction algorithm and a method for computing leaf payoffs of truncated games. This allowed their player to achieve the third place in the series competition at the 2006 AAAI Computer Poker Competition [54]. However, several disadvantages were identified [2]. First, solving the linear programming in real-time is not a task that can, currently, be computed quickly as intended. Second, because of design decisions, GS2 separated the game into two phases, early and late, which prevented it from having an accurate estimate of the utility of certain lines of play.

Their latest autonomous poker agent, the GS3 [40], introduced a new abstraction algorithm for sequential imperfect information games [58] as well as abstract and game-theoretically analyze all four betting rounds in one run, rather than splitting the game into phases [40]. This new approach led them to a second place in no-limit and the third place in the limit Texas Hold'em at the 2007 AAAI Computer Poker Competition [56].

3.4.6 Smallbot Family

Smallbot 1239, 1399 and 2298 are ε-Nash equilibria strategies produced by the University of Alberta CPRG. Their names come from their generation number and number of iterations of the applied algorithm.

Smallbot family bots are based on a published technique called Range of Skill. This method does not directly involve solving a linear program. The main idea is to create a sequence of agents, where each agent can defeat the previous agents in the sequence by at least ε. For any game and value for ε, eventually, the sequence approaches within ε of Nash equilibrium, and no further agents are able to defeat it by more than ε [30]. Also, this technique allowed for a consistent, whole-game strategy to be created instead of overlapping strategies than split the game into different phases.

Smallbot 1239 and 1399 are considered weaker than PsOpti4 poker agent [30] but Smallbot 2298 performed quite well when tested, even beating all competitors present in 2006 AAAI Computer Poker Competition.

3.4.7 Vexbot

Vexbot [23] was the first successful adaptive player to be created by the University of Alberta CPRG. This artificial player‘s strategy core was to build an opponent model online (while playing) and then, using the miximix search technique, calculate strategies and exploit the opponent model effectively.

To be effective, two types of information were provided by the model: opponent action probabilities at their choice nodes, and the utility for reaching each terminal node. The latter was determined by measuring the frequency of the opponent‘s actions for each betting history. The former is the product of the size of the pot and the probability of winning. [30]

The ability to adapt their play by learning from the opponent was a great advantage at the time, achieving much success both against human and computer players [23], and an important breakthrough in the construction of a world class poker player. Unfortunately it featured several disadvantages [25]. One of these was at the start of a match, at a stage where the program didn‘t know the rules of poker and could develop impossible beliefs about the game [30]. Therefore the requirement for solid effective default data was crucial. Another one was the substantial amount of hands required to be played before an effective opponent model could be generated [25].

Noteworthy is the fact that this program was licensed into the commercial software trainer Poker Academy.

3.4.8 BRPlayer

BRPlayer is the successor to Vexbot. It was named so since it plays a best-response strategy. Like its predecessor, BRPlayer doesn‘t employ a static strategy. It records observations about its opponents’ actions, and develops a model of their style of play. It continually refines its model during play and uses this knowledge of the opponent to try to exploit his weaknesses.

Both of these programs share the same action-selection search procedure, mininix. They differ only in the type of opponent model used. Both the BRPlayer and Vexbot use a context tree for modeling their opponent's action frequencies and they both assume chance node outcomes occur with uniform probability. The difference between the two player's models lies in how they model their opponent's showdown information. [28]

Although an improvement, BRPlayer suffered from most of the same disadvantages found in Vexbot [30]. Experiments to test the BRPlayer's performance against human competition were never setup but BRPlayer was a major component in the University of Alberta poker program, Poki-X, in 2005 World Poker Robot Championship [59] match against poker professional player, Phil Laak.

3.4.9 Polaris

Polaris is a set of different poker agents that featured in the Man vs. Machine contest promoted by the University of Alberta. This particular contest of limit heads-up Texas Hold'em opposes several human players against the computer poker player developed by the CPRG.

The first version of Polaris was comprised of agents that used ε-Nash equilibrium, restricted Nash response and an experimental aggressive ε-Nash equilibrium strategy [30]. This version featured in 2007's first edition of the Man vs. Machine contest against two professional poker players. Although not victorious in the contest [60], post-game analysis and player comments about Polaris led the research team conclude that poker agents were quickly approaching world champion levels of play in this game [30].

The second version of Polaris, named Polaris II, featured in the second edition of the Man vs. Machine contest, in 2008. This time six professional poker players were invited to outmatch the computer player in a six match game. Polaris performance was the best ever and at the end of the competition [61] Polaris became known as the first artificial poker player to beat a team of professional human players. This version of Polaris was subject to much improvement from the previous version and featured a recently published [62] new technique, based on importance sampling, which greatly improved the accuracy of its estimators.

Chapter 4

4.Poker Learner

Poker Learner is the name given to the agents developed in this work. At total, four agents were developed with the same general structure, but different implementations.

4.1 General Architecture

The architecture of the Poker Learner is divided into several modules, interacting with each other to create an information flow consistent with the underlying framework.

The modules contained by the agents consist of:

- An Opponent Modeling module: to be able to identify an opponent and to learn the best actions against each of them.
- A State module: since a Reinforcement Learning it's being used, it is necessary to map the environment of the game. Also used to choose the best action for each state and to update reward value of the chosen actions.
- A Hand Evaluator module: needed to compare own cards with opponent cards.
- GameInfo module: given by the framework. It contains all the information related to the current game, players and actions.

More details about each module will be given in the subsections ahead about implementation.

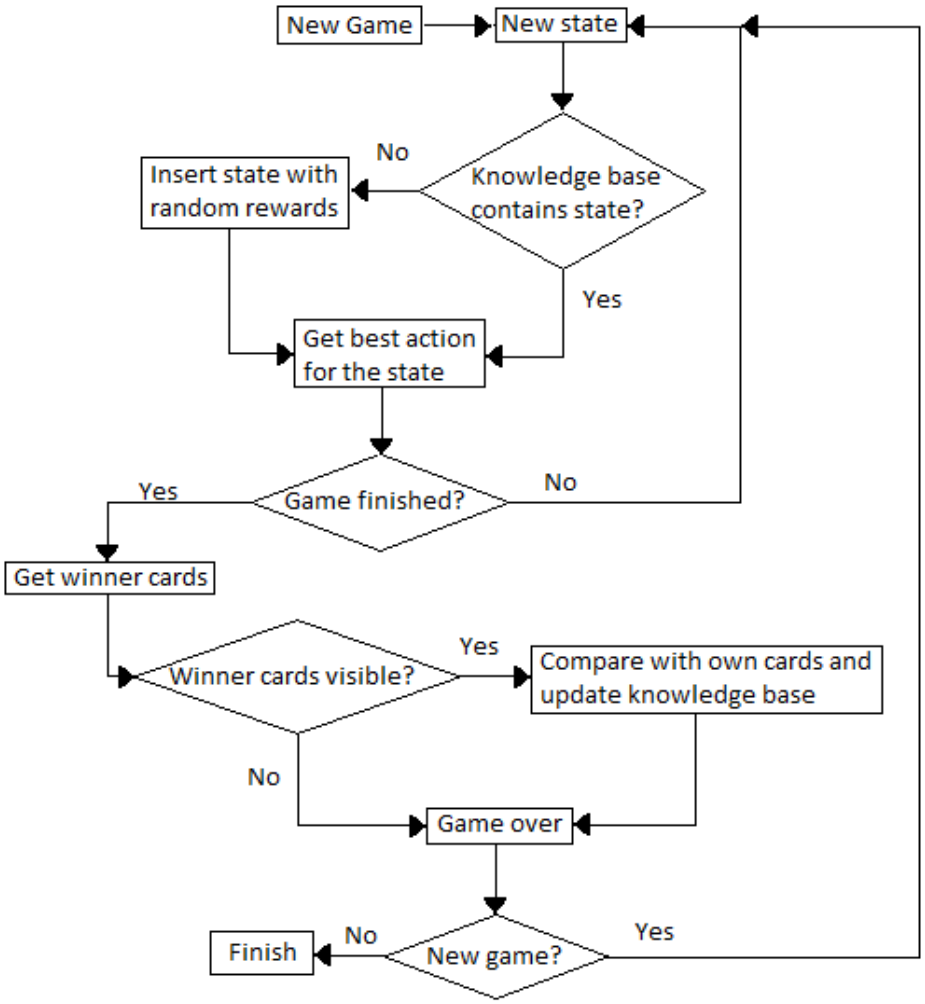

Figure 4.1 Diagram of Poker Learner's structure

As seen in Figure 4.1, the agent's structure is rather simple to engage in the complexity of the game.

When the player is sent into a Poker table for the first time, its Knowledge Base is completely empty. This is to allow the agent to learn by itself, without any interference from outside.

The Knowledge Base is first updated when the player receives its hole cards and has to decide which action to choose. The agent will analyze the game variables and join together the variables that describe the state of the game. Then it will search the Knowledge Base for the state, but since the Knowledge Base is empty, the agent has to insert the new state into it, and randomly fill the reward values for the actions. This is to allow the agent to have a dynamic start and avoid being modeled in the beginning.

Since it now has the current state filled, it's time to choose what it considers the best action for the state.

After the agent's action, the game can either move to the Flop stage (if all players call or fold) or maintain in the Pre-Flop stage (if at least one player raised the bet after the agent). If the game continues in Pre-Flop, the agent will have to analyze again the game variables to build the new state of the game and question the Knowledge base about this new state. On the other hand, if the game goes into the Flop stage and the agent hasn't fold yet, the agent goes all-in, so that the game ends and goes directly to showdown.

During the showdown, the player will compare its cards with any player who has also gotten to the showdown (also comparing with itself). After the comparison, the action chosen receives a reward based on how well chosen it was – to choose a fold when the opponent players had a higher hand will give a positive reward to fold for that state.

The game ended here, but there is always a possibility of a new game. In case this was the last game of the series, the agent storages its Knowledge Base in an external file, so that the next time it is sent to a Poker table it doesn't have to learn everything from zero again.

4.2 Implementation

4.2.1 Modules

4.2.1.1 GameInfo

The GameInfo module is an object of the Open Meerkat framework which contains all the information about the game. Some of the more useful information is:

- Number of players: this information is used to locate the agent's position in the table, and also to verify if it is worthy of applying a learning strategy (due to some restrictions in the learning method).
- Board cards: the board cards are used when it is necessary to compare the agent's 5-card hand with an opponent's hand.
- Actions taken: the actions are used to build an environment state for the Knowledge Base. Also, the Opponent Modeling module wouldn't be useful if this information was not available.
- Player information: the player information has several utilities. First, it is used for the Opponent Modeling module, either through the actions taken by the player or by the games it played. Secondly, it is used to obtain the player's hole cards, needed to compare with the agents own cards and to verify accuracy of chosen action.

- Game stage: this information is needed to decide the strategy to use, although since the agents only play in Pre-Flop, it's necessary to know whether they use their strategy or go all-in.
- Amount to Call: the amount to call is the amount of the last bet made, so the agent needs this information to know how much it will put to the pot if it called or raised the bet.

4.2.1.2 Opponent Modeling – PlayerInfo class

The Opponent Modeling module is used to identify different players and to create a strategy against their playing style.

During this work, two approaches were made to Opponent Modeling, one was to identify the players by their names, and the other was by their playing style.


```
private class PlayerInfo {
    long ngames;
    long nflops;
    long ncalls;
    long nraises;
    int lastAction;

    public PlayerInfo() {
        ngames = 0;
        nflops = 0;
        ncalls = 0;
        nraises = 0;
    }
}

private static int playerType(long ngames, long nflops, long ncalls, long nraises) {
    double AF = 1;
    if (ncalls != 0)
        AF = nraises / (ncalls / 1.0);
    double Tightness = 1;
    if (ngames != 0)
        Tightness = nflops / (ngames / 1.0);

    if (AF > 1) {
        if (Tightness <= 28) {
            return TightAgressive;
        } else {
            return LooseAgressive;
        }
    } else {
        if (Tightness <= 28) {
            return TightPassive;
        } else {
            return LoosePassive;
        }
    }
}
```

Figure 4.2 Opponent Modeling for player types

The Opponent Modeling for the player's playing style consists of a class to contain the information needed, and a function to convert that information into a classification. This classification is based on the Aggressive Factor of the player and it's Tightness, and these two factors are calculated using the information stored in the PlayerInfo class. The classification is given in an integer:

- 0 – Tight Aggressive;
- 1 – Tight Passive;
- 2 – Loose Aggressive;
- 3 – Loose Passive.

```
class Players {
    String name = "None";
    int Action;

    public Players(String name) {
        this.name = name;
    }
}
```

Figure 4.3 Opponent Modeling class for Name Learner

As it is showed in Figure 4.3, the Opponent Modeling class for the name identification is simple; it contains only the name of the player and the last action it performed.

The player's name is inserted when such player does not exist in the agent's memory base. The player's action is updated every time the player calls or raises the bet.

This approach was to solve the problem of having the first few games to accurately calculate the opponent's playing style.

4.2.1.3 Hand Evaluator

To know if the chosen action was the correct one, the hand of the agent needs to be compared with the ones of the opponents.

There were two methods used for comparing hands.

The first one was to compare both players' best five-card combination. To do this, it was used a hand evaluator imbued in the Open Meerkat framework done by Steve Brecher [63], which receives a seven-card hand and returns an integer corresponding to the best five-card rank of that hand.

The second method was to compare both players' hole cards. Since there were only two cards in each hand, it was only needed to compare the ranks of each card, without the need for an external hand evaluator.

```
int temp = 0;
if (card1.getRank() == card2.getRank()) {
    if (this.c1.getRank() != this.c2.getRank()) { temp--; }
    else
        if (card1.getRank() > this.c1.getRank()) { temp--; }
        else
            if (card1.getRank() < this.c1.getRank()) { temp++; }
} else {
    if (this.c1.getRank() == this.c2.getRank()) { temp++; }
    else
        if (card1.getRank() > card2.getRank()) {
            if (card1.getRank() > this.c1.getRank()) { temp--; }
            else
                if (card1.getRank() < this.c1.getRank()) { temp++; }
                else
                    if (card2.getRank() < this.c2.getRank()) { temp--; }
                    else
                        if (card2.getRank() < this.c2.getRank()) { temp++; }
        } else {
            if (card2.getRank() > this.c1.getRank()) { temp--; }
            else
                if (card2.getRank() < this.c1.getRank()) { temp++; }
                else
                    if (card1.getRank() < this.c2.getRank()) { temp--; }
                    else
                        if (card1.getRank() < this.c2.getRank()) { temp++; }
    }
}
```

Figure 4.4 Hole Cards comparison

First, there was a need for a control variable – "temp" – to be adjusted by the results of the hands comparison, then the cards for both hands – "c1" and "c2" for the agent and "card1" and "card2" for the opponent.

The first step of the comparison is to check if the opponent has a Pair. If this is the case, then it's time to check the agent's hand. Three scenarios can happen:

- The agent only has a High Card: since a High Cards scores lower than a Pair, the control variable is decremented;
- The agent has a lower ranking Pair: the control variable is decremented;
- The agent has a higher ranking Pair: the control variable is incremented.

If the opponent only has a High Card, several situations arise for the agent's hand, and the outcome of the comparison:

- A Pair: the control variable is incremented due to the fact that the Pair scores higher than the opponent's High Card;
- Lower ranking higher card: the higher ranking card of the opponent is higher than the agent's so the control variable is decremented;
- Higher ranking higher card: the opposite of the point above, the control variable is incremented.
- Same higher card but lower second card: since it is a tie with the hand's higher card, the second card can break the tie. In this case the agent loses and the control variable is decremented;
- Same higher card but higher second card: the same logic of the above point applies here. The control variable is incremented.

When the agent and the opponent have a similar hand, they are tied and the control variable does not change its value.

At the end of the comparison the control variable can have one of three values:

- Lower than zero: the agent's hand is ranked lower than the opponent's;
- Zero: both players have similar hands and are tied;
- Higher than zero: the agent's hand is ranked higher than the opponent's.

4.2.1.4 State

The State module controls the Knowledge base of the agent. While the remaining modules are functions within a class, the State module is a container to keep and update the information of each state, while the methods to decide what and how to update are implemented on another method of the agent.

As shown in Figure 4.5, the information contained in the module refers to the variables used to construct the state of the environment and the values for the actions available in that state.

While the agents have different types of variables for the state, there is a common branch:

- Hand value: a value is attributed to a group of card combinations with the same rank. The higher the value, the higher the probability of winning a game with that combination in Pre-Flop;
- Button position: the position of the agent in reference to the dealer. While the number of players for the Open Meerkat framework can vary from two to six, the

values for the button position were diminished to three in the agent – early position, mid position and late position;

- Last Player: information on the last player who called or raised the pot. Can either be its playing style or its name;
- Last Player action: value representing the action of the last player. The action is either call or raise.

```
private class State {
    int handValue;
    int buttonPos;
    int lastPlayer;
    int lastPlayerAction;
    int fold;
    int call;
    int raise;
    int nCases;
    int hsLearning;
    int whLearning;

    public State(int hv, int bp, int lp, int lpa) {
        this.handValue = hv;
        this.buttonPos = bp;
        this.lastPlayer = lp;
        this.lastPlayerAction = lpa;
        this.fold = (int) (Math.random() * 100);
        this.call = (int) (Math.random() * 100);
        this.raise = (int) (Math.random() * 100);
        this.nCases = 0;
        this.hsLearning = (int) (Math.random() * 100);
        this.whLearning = (int) (Math.random() * 100);
    }
```

Figure 4.5 State structure

The actions available for each state are the Poker betting actions available: fold, call, and raise. Also in two of the agents there is also the possibility of choosing the learning method.

There is also another variable in the state that was not mentioned until now, "ncases". This variable is only a reference to the number of times that the values of the state actions were updated.

The state variable need to be inicialized with the concrete information, i.e., there can't be a state where one of its variables is unknown. On the other hand, the action variables are initialized with a random number between 0 and 99 since the correct values are unknown. The control variable "ncases" is initialized at zero.

Also in this module there is the selection and update of the state actions.

The update of the actions is based on six rules:

- Increment Fold: increment the value of Fold and decrement the value of Call and Raise;
- Decrement Fold: decrement the value of Fold two times; the values of Call and Raise are incremented normally;
- Increment Call: increment the value of Call, increment the value of Raise two times and decrement the value of Fold;
- Decrement Call: decrement Call two times, decrement the value of Raise four times and increment Fold;
- Increment Raise: increment the value of Raise and decrement the values of both Call and Fold;
- Decrement Raise: decrement the value of Raise two times and increment the values of Call and Fold.

The choice of the action can be made by two methods, choosing the action with the higher value or choosing randomly, taking into consideration their values.

```
public int getAction() {
    double temp = Math.random();
    double f, c;
    f = (this.fold * 1.0) / ((this.fold + this.call + this.raise) * 1.0);
    c = (this.call * 1.0) / ((this.fold + this.call + this.raise) * 1.0);
    if (temp <= f) {
        return Action.FOLD;
    } else {
        if (temp <= (f + c)) {
            return Action.CALL;
        } else {
            return Action.RAISE;
        }
    }
}
```

Figure 4.6 Method to randomly choose the action to take

The values for the learning methods are based on the number of incorrect choices made during the game, i.e., if an agent chooses to raise a pot with cards lower than the opponents, aside from getting a negative reward for the action of raise, it will also receive a negative reward for the learning method for that state.

4.2.2 Agents

Four agents were developed during this work. Each of them represents an approach to develop a Reinforcement Learning agent for Pre-Flop Texas Hold'em Poker.

4.2.2.1 Hole Cards Learner – HCLearner

This agent is the most simple of the four developed. Its learning method is based on the two hole cards in its possession, and comparing them with the opponents to verify the accuracy of its choice and update the actions values. The Opponent Modeling implemented is based on opponent playing style and the action chosen is based on its values.

This was the first approach to the problem, trying to maintain the agent as simple and generic as possible.

4.2.2.2 Winning Hand Learner – WHLearner

This agent is a modification of the one above. The difference between the two is the learning method; this one verifies the accuracy of its actions by comparing its best five-card hand with the winner of the game.

The second approach to the problem. The objective with this agent was to ascertain the link between the two hole cards and the final hand, and how that would interfere in the results of the agent.

4.2.2.3 Name Learner

This agent is a little less generic than the two above, and is also a little more complex. The opponent information was altered from playing style to player's name, and the learning method was changed to cover both of the approaches. Also, the choice of the action is now random.

The objective of this agent was to verify the possibility of having the agent deciding which one of the learning methods was better and for what case. Also, the decision the change the information of the opponent players was made to avoid the possible mistakes the other agents made with their choices while trying to discover the opponents playing style.

4.2.2.4 Poker Learner

This agent was developed using the knowledge obtained from all of the previous agents. The Opponent Modeling is based on playing style, and the learning and action choosing method used are the same as the ones from the Name Learner.

The objective of this agent was to develop the final product and verify if by joining the best components of each of the other agents this agent would give better results.

Chapter 5

5.Experiments and Results

5.1 Experiments

To test the developed agents and to verify their results it is necessary to have them play against other poker playing bots. For this experiment, four bots were chosen to compare against:

- **AlwaysAllInBot**: a poker playing bot that the only action it makes is to bet all of its money on any hand. This agent is considered a Loose Aggressive player.
- **AlwaysCallBot**: similar to the agent above, but this one only calls the bets made. This player is described as a Loose Passive.
- **HandStrenghtBot**: a bot that takes its hand strength [64] into consideration for its betting strategy.
- **SimpleBot**: a complex, but efficient poker bot. Contains several generic rules to play in Pre-Flop, also contains a simple implementation of opponent modeling. It is one of the agents included in the Open Meerkat Testbed [49]. To avoid waisting unnecessary computational power it was changed to play only in the Pre-Flop stage.

The experiments will be made in the Open Meerkat Testbed, in a mode where negative bankroll is permited so that it is possible to better analyse the results of the agent's learning. There will be two phases of experiments:

The first will test the agents learning methods. Each agent will face five opponents of the same type on a series of games. It is expected that when the values for the state actions have stabilized, the agent has developed the best strategy for that type of opponent.

The second phase will test how well the strategies work against the opponents. In this phase, the agents will face only one opponent, a one-on-one match. It is expected that each agent manages to win, or at least, contain its losses at a minimum when facing the opponents in this phase.

Aside of going against only one type of opponent at a time, there will also be another test case where each agent will face all five opponents in the same table. This will test the agent's performance in a table with several opponents.

There are two types of results that need to be analyzed, the first are the results of the games against the opposing agents, with this their efficiency against other basic poker playing agents will be tested and show how well they can manage themselves in a table with several opponents.

The second result is the one from the learning the agents will be doing. Although it is more difficult to analyze and accurately see faults in every state, it is possible to verify some of the more extreme cases as a 2-3 hand or an Ace-Ace hand.

5.2 Game Results

Since these games will be in a two-player variant and both start with 0$ as bankroll, the ones with a positive bankroll at the end of the games are the winners. The only exception is the competition where the agent's position will also be mentioned.

5.2.1 Matches against AlwaysAllInBot

The AlwaysAllInBot is a static agent, so the Learner agents could easily learn a counter-strategy to beat this bot.

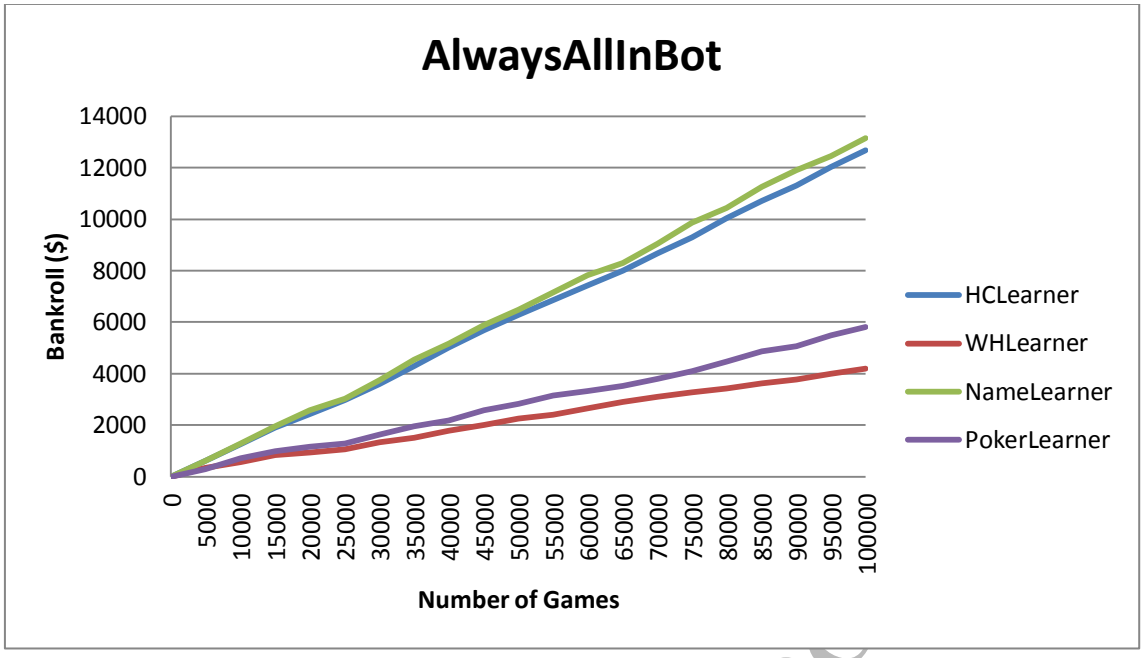

Figure 5.1 Bankroll evolution against AlwaysAllInBot

It can be observed in Figure 5.1 that all agents managed to win against this bot, although both Poker Learner and Winning Hand Learner had a lower performance than the remaining two agents. Also, the graph is smooth, meaning that there weren't any big turnaround during the games, so the more games played by the agents the higher their winnings would be. Table 5.1 shows the precise bankroll the agents had for the start middle and end of the game.

Table 5.1 Bankroll against AlwaysAllInBot

Number of games	Hole Cards Learner	Winning Hand Learner	Name Learner	Poker Learner
5000	606.97$	324.76$	605.48$	292.09$
50000	6285.92$	2246.17$	6488.91$	2831.04$
100000	12669.82$	4187.45$	13158.19$	5801.28$

5.2.2 Matches against AlwaysCallBot

The expected results for the game against the AlwaysCallBot are similar to the ones of the AlwaysAllInBot, mainly because of the similarities between the two agents.

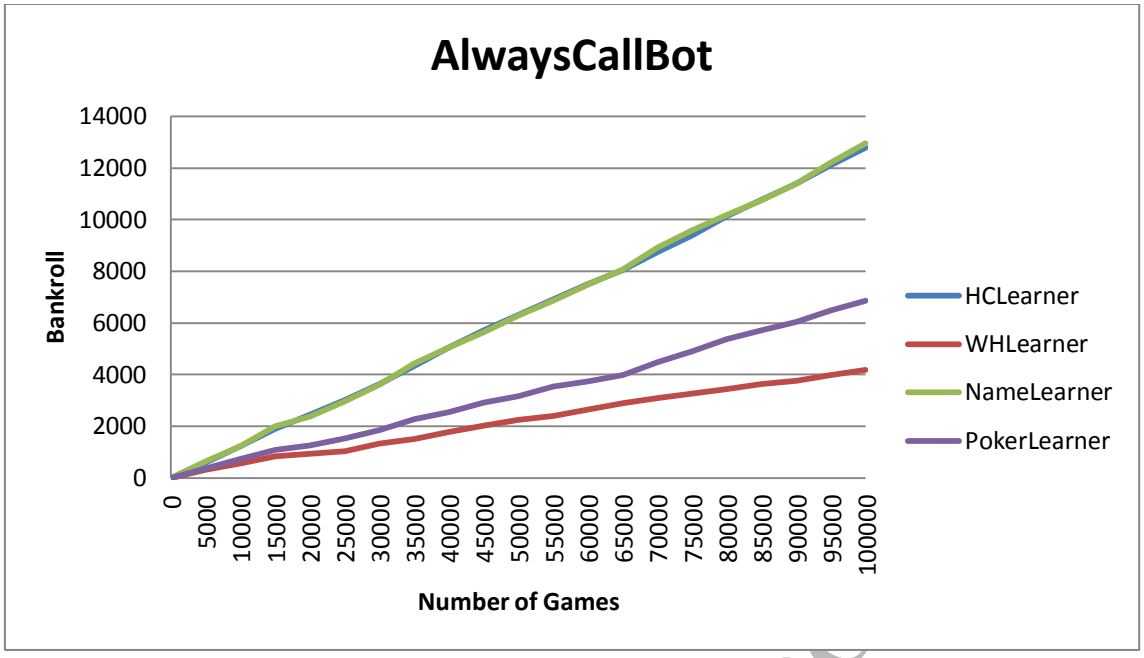

Figure 5.2 Bankroll evolution against AlwaysCallBot

Again all agents managed to win against opposing agent, being the results not too different from the results against the AlwaysAllInBot. Also, from Figure 5.2 it is possible to observe that the Poker Learner had a slight better result this time.

Table 5.2. Bankroll against AlwaysCallBot

Number of games	Hole Cards Learner	Winning Hand Learner	Name Learner	Poker Learner
5000	588.68$	324.76$	640.52$	373.76$
50000	6325.80$	2246.17$	6290.70$	3178.86$
100000	12782.77$	4187.45$	12944.63$	6856.90$

5.2.3 Matches against HandStrengthBot

The HandStrengthBot has a static strategy; however it is a better strategy than the ones of the two agents described above. For this reason, while not expecting to have a negative bankroll at the end of the game, it is expected that the amount would be inferior.

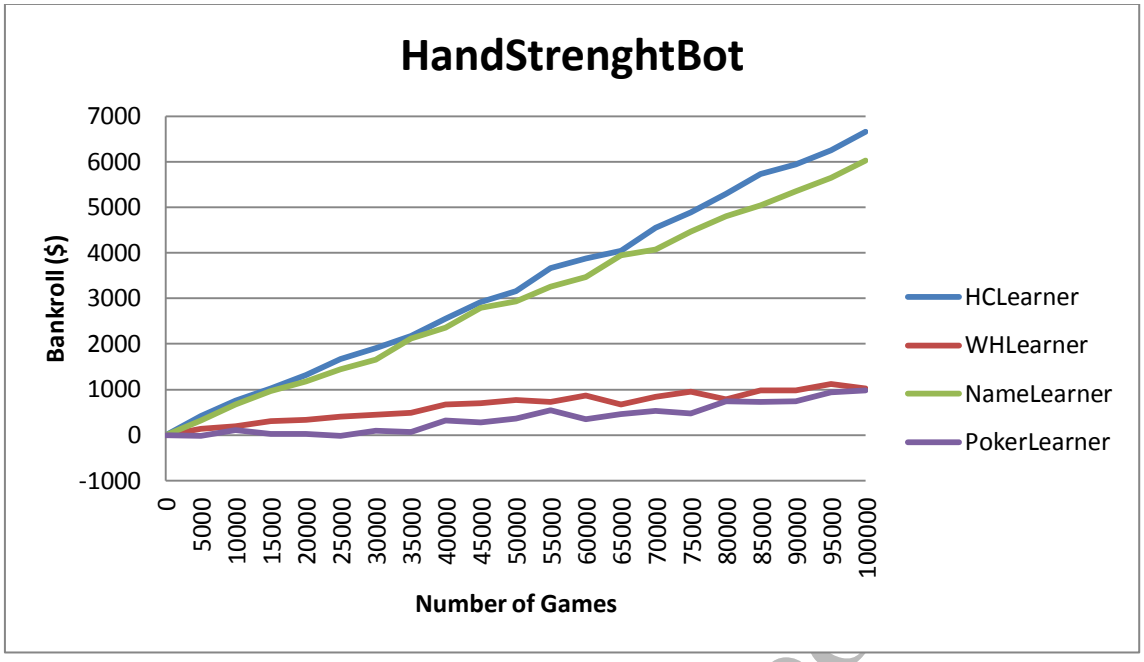

Figure 5.3. Bankroll evolution against HandStrenghtBot

As it was expected, the final amount of each agent was lower, and Poker Learner even had a negative bankroll for a few games. Even so, the results show that these agents can find the weaknesses in the opponent's strategies and take advantage of it.

Table 5.3. Bankroll against HandStrenghtBot

Number of games	Hole Cards Learner	Winning Hand Learner	Name Learner	Poker Learner
5000	411.58$	130.46$	313.57$	-18.23$
50000	3160.99$	770.21$	2932.19$	354.98$
100000	6657.04$	1015.85$	6029.68$	982.55$

5.2.4 Matches against SimpleBot

This agent is the true test of the two-player games for the Learner agents. Despite being called "simple", its strategy is not simple at all and the agent can play at an intermediate level.

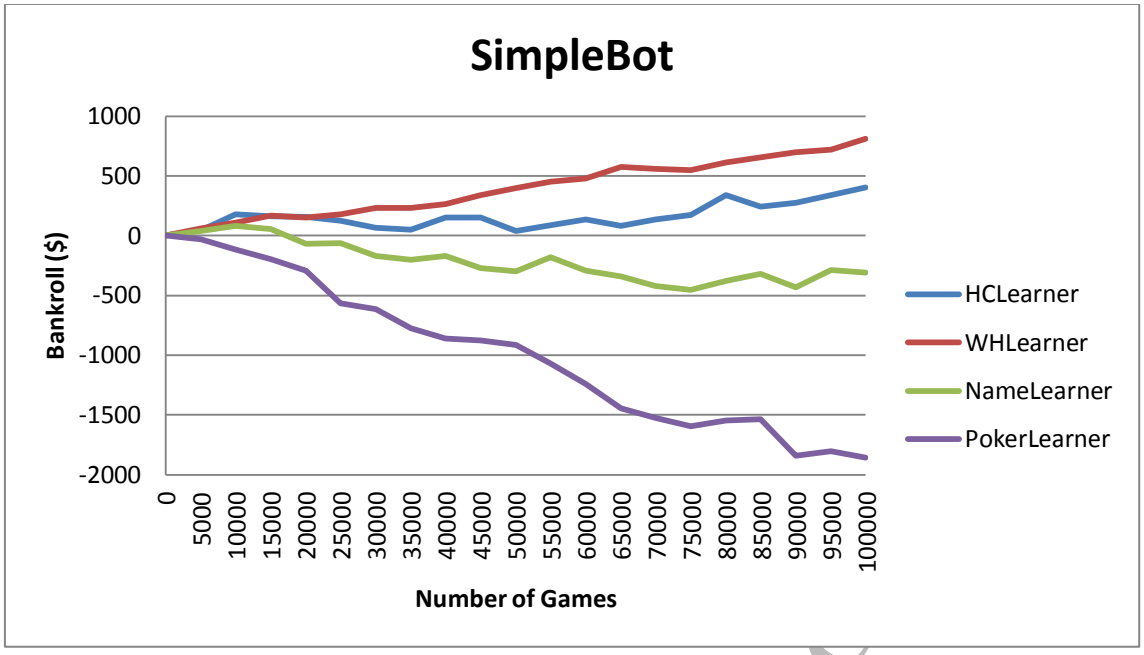

Figure 5.4. Bankroll evolution against SimpleBot

As it can be seen in Figure 5.4, both Name Learner and Poker Learner got negative results. Although, it is worth mentioning that the Winning Hand Learner managed to stay somewhat unaffected with these results since it was the one with the lower difference with the other opponent bots. One possible explanation for these results is the methods used to accelerate the learning, while the lerning rate is faster, itmakes it easier for opponents to model the agents strategy. It is also possible to see some turn around in the game development, especially with the Hole Cards Learner, as it wins and loses games.

Table 5.4. Bankroll against SimpleBot

Number of games	Hole Cards Learner	Winning Hand Learner	Name Learner	Poker Learner
5000	47.89$	60.53$	40.39$	-29.18$
50000	37.68$	397.10$	-299.80$	-915.64$
100000	404.57$	812.05$	-309.21$	-1858.30$

5.2.5 Competition

Although the results in the one-on-one games are promising, most games of Poker are played with a multi-player table. This way, it is possible to exploit the weakness of a weaker opponent to win against a thougher one.

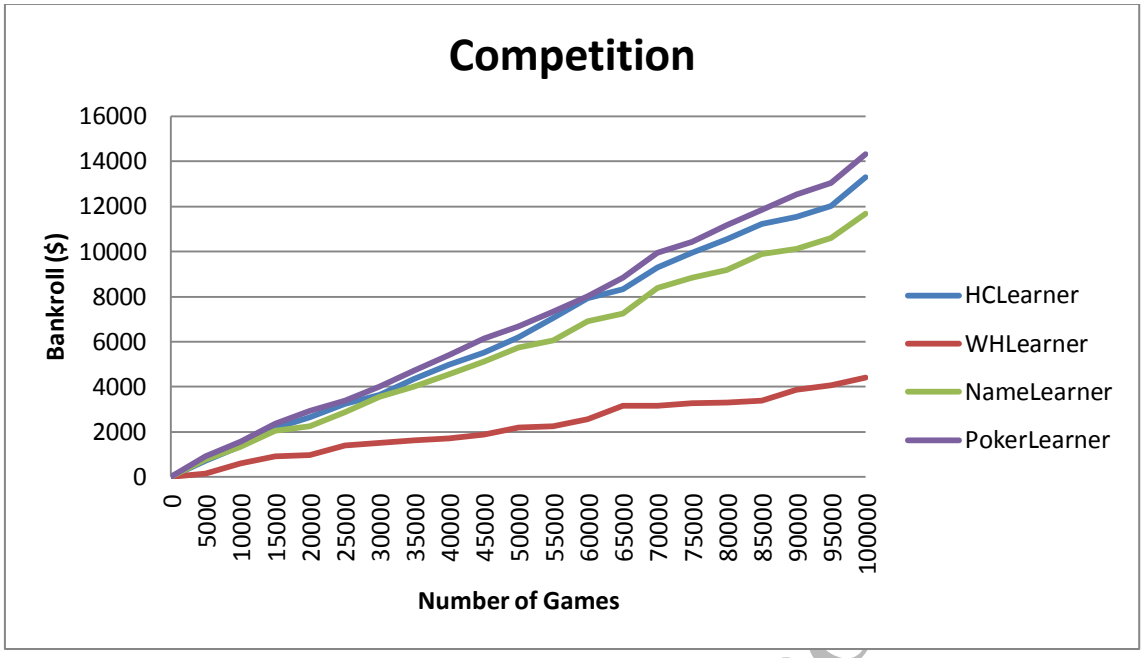

Figure 5.5. Bankroll evolution in a competition

As it is possible to see, all the agents had good results, except for the Winning Hand Learner, which placed second on its competition. This result sort of contradicts the results from the earlier experiments, since, for example, the Poker Learner had a negative result against the SimpleBot, but here it managed to not only win against the SimpleBot and the remaining opponents, but also to have the highest bankroll of all of the Learner agents.

5.3 Learning Results

Despite the game results beeing important, so are the learning results. To be able to compare the results between all the agents, all of them made 500000 games, althought most of the agents were winning around the 100000 games. Now there will be presented some cases where it is possible to correctly verify the accuracy of the learning methods.

5.3.1 Hole Card Learner

Table 5.5 Learning results for Hole Card Learner

Hand Value	Button Position	Last Player	Last Player Action	Fold	Call	Raise
Unsuited 7-2	Late	Tight Passive	Raise	82%	0%	18%
Unsuited 5-3	Late	Tight Agressive	Raise	94%	0%	6%
Suited A-7	Late	Tight Passive	Call	0%	2%	98%
Pair of 8s	Late	Tight Agressive	Raise	4%	0%	96%

Despite not being expected to have the perfect strategy, the agent managed to create a reasonable approximation, although some values can be considered a little out of place, such as a combination of unsuited 7-2 cards having such a high rate for Raise while having zero for Call.

5.3.2 Winning Hand Learner

Table 5.6 Learning results for the Winning Hand Learner

Hand Value	Button Position	Last Player	Last Player Action	Fold	Call	Raise
Suited 3-2	Mid	Tight Passive	Call	98%	2%	0%
Unsuited 6-2	Early	Tight Aggressive	Raise	90%	10%	0%
Suited A-Q	Late	Tight Aggressive	Raise	91%	9%	0%
Pair of Aces	Mid	Tight Passive	Raise	0%	1%	99%

In Table 5.6, it can clearly be seen a fault in the learning system. The agent assumes that a suited hand of Ace-Queen is a bad hand and needs to fold, while in reality it is a good hand and it should have gone to the Flop stage.

5.3.3 Name Learner

Table 5.7. Learning results for the Name Learner

Hand Value	Button Position	Last Player	Last Player Action	Fold	Call	Raise
Suited 6-2	Late	HandStrenght Bot	Raise	98%	1%	1%
Unsuited 5-2	Late	SimpleBot	Call	100%	0%	0%
Pair of Kings	Late	SimpleBot	Raise	26%	31%	43%
Pair of 10s	Late	SimpleBot	Call	0%	4%	96%

Despite not being entirely wrong this agent also has some minor miscalculations on some hands. One possible reason could be that, in some of the games, the opponent had a better hand, or maibe there werent's enough test cases for that state.

5.3.4 Poker Learner

In the cases taken, this agent took extreme positions in relation to the action to take. This may be from a high number of test cases, where the the numbe of wrong choices are neglitible.

Table 5.8 Learning results for the Poker Learner

Hand Value	Button Position	Last Player	Last Player Action	Fold	Call	Raise
Suited 3-2	Early	Tight Aggressive	Call	100%	0%	0%
Suited 5-2	Early	Tight Aggressive	Raise	100%	0%	0%
Pair of 10s	Early	Tight Passive	Call	0%	0%	100%
Pair of Kings	Early	Tight Aggressive	Raise	0%	0%	100%

Chapter 6

6.Conclusions

While writing the goals for this dissertation project, it was believed that the resulting agents couldn't win against every poker bot ever developed. Although that being true, the developed agents showed some promising results, playing against simple agents, and winning by a rather large margin.

The achievements of the Hole Cards Learner came as a good surprise as it was the simplest of all the Learner agents, and it was the one with the better performance of all of them.

The Winning Hand Learner, developed with the Flop stage in mind, didn't perform so well, aside from the game against the SimpleBot. The possible reason for that occurrence was because of the fact that the two hole cards by themselves can't guarantee the victory in Texas Hold'em. This means that the final hand may not be entirely related to the player's hole cards, making it harder for the SimpleBot to model the agent.

The Name Learner met the expectations, but while it was faster to identify the opponent, it also became less dynamic, wich led to its defeat against the SimpleBot.

The Poker Learner, despite having a rather high expectation in the beginning, it only came to prove that once again good plus good isn't twice as good. But while it didn't excel in the two-player games, it did manage to have the highest bankroll of the competition between learner agents.

In the competition all agents proved that they could take advantage of one opponent weakness to win another opponent that otherwise would be too hard to defeat.

Although this dissertation project is over, there are many improvements that can be done to the Learner agents in the future. One improvement that could take this agent to a higher level could be the development of the strategy for the Post-Flop stages of the game. Aside from this

one, there are other smaller improvements that could make big changes, such as improving the opponent modeling module, make a better description of the state of the environment, and others.

Chapter 7

7.References

[1] John McCarthy. What is Artificial Intelligence?, November 2007. http://www-formal. stanford.edu/jmc/whatisai/whatisai.html (cited: 31st May, 2011).

[2] Pamela McCorduck. *Machines Who Think*. A. K. Peters, Second Ed., 2004.

[3] Mary Shelley. Frankenstein. Harding, Mavor & Jones, 1818.

[4] James M. White, Gene D. Rohrer. Image Thresholding for Optical Character Recognition and Applications Requiring Character Image Extraction. In: IBM Journal of Research and Development, vol. 27, no. 4, pp. 400-411.1983.

[5] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. Unpublished manuscript. *International Conference on Computer Vision and Pattern Recognition*, 2011.

[6] Monty Newborn. *Kasparov Versus Deep Blue: Computer Chess Comes of Age*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[7] David Ferrucci, Eric Brown, Jenifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, Chris Welty. Building Watson: An Overview of the DeepQA Project. In: *The AI magazine*, vol. 31, no. 3, pp. 59-79. American Association for Artificial Intelligence, Menlo Park, CA, USA, 2010.

[8] Online Poker Industry History & Growth. http://online-poker.flopturnriver.com/ (cited: 3rd June, 2011).

[9] The Poker Channel Europe. http://www.pokerchanneleurope.com/ (cited: 3rd June, 2011).

[10] Charles Livingstone. The social economy of poker machine Gambling in Victoria. In: *International Gambling Studies*, vol. 1, no. 1, pp. 46-65. Routledge, 2001.

[11] Paul H. Delfabbro, Anthony H. Winefield. Poker-machine gambling: An analysis of within session characteristics. In: British Journal of psychology, vol. 90, no. 3, pp. 425-439. British Psychological Society, Leicester, 1999.

[12] PokerNews. http://www.pokernews.com/pokerterms/as-nas.html (cited: 7th June, 2011).

[13] David Parlett. Card Games: History of Poker, 2005. http://www.pagat.com/poker/history.html (cited: 7th June, 2011).

[14] John McLeod. Rules of Card Games: Pochspiel, 27th September, 2005. http://www.pagat.com/stops/poch.html (cited: 8th June, 2011).

[15] James Hildreth. *Dragoon Campaigns to the Rocky Mountains: A History of the Enlistment, Organization and First Campaigns of the Regiment of U.S. Dragoons 1836*, Kessinger Publishing LLC, 2005.

[16] Joe Cowell. *Thirty Years Passed Among The Players in England and America*. Harper & brothers, 1844.

[17] Jonathan Green. *An Exposure of the Arts and Miseries of Gambling*. Redding, 1845.

[18] Henry Bohn. *Bohn's New Handbook of Games*. Henry Anners, 1850.

[19] Adel Awwad. The History of 7 Card Stud Poker, 6th February 2005. http://ezinearticles.com/?The-History-of-7-Card-Stud-Poker&id=111152 (cited: 9th June, 2011).

[20] John Dahl. Rounders (movie), 1998.

[21] Roy Cooke, John Bond. *Cooke's Rules of Real Poker*. ConJelCo, 2005.

[22] Kelli Mix. *The Game Day Poker Almanac Official Rules of Poker*. Flying Pen Press LLC, 2007.

[23] Darse Billings, Aaron Davidson, Terence Schauenberg, Neil Burch, Michael Bowling, Robert Holte, Jonathan Schaeffer, Duane Szafron. Game-Tree Search with Adaptation in Stochastic Imperfect-Information Games. In Lecture Notes in Computer Science: Computers and Games, vol. 3846, pp. 21-34. Springer Berlin, Heidelberg, 2006.

[24] Aaron Davidson. Using Artificial Neural Networks to Model Opponents in Texas Hold'em. Unpublished manuscript. http://spaz.ca/aaron/poker/nnpoker.pdf, 1999.

[25] Darse Billings. *Algorithms and Assessment in Computer Poker*. Ph.D. thesis. University of Alberta, 2006.

[26] Scott Schumacher. Probabilistic Versus Deterministic Data Matching: Making an Accurate Decision, January, 2007. http://www.information-management.com/specialreports/20070118/1071712-1.html (cited: 15th June, 2011).

[27] Aaron Davidson. *Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment*. Ms. C. thesis. University of Alberta, 2002.

[28] Terence Schauenberg. *Opponent Modeling and Search in Poker*. Ms. C. thesis. University of Alberta, 2006.

[29] Ian Frank, David Basin, Hitoshi Matsubara. Finding Optimal Strategies for Imperfect Information Games. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI'98/IAAI'98, pp. 500-507. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1998.

[30] Michael Johanson. *Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player*. Ms. C. thesis. University of Alberta, 2007.

[31] Frans Oliehoek. *Game theory and AI: a unified approach to poker games*. Ms. C. thesis. University of Amesterdam, 2005.

[32] Andrew Gilpin, Tuomas Sandholm. 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*, vol. 2, pp. 1007-1013, 2006.

[33] Teppo Salonen. BluffBot – Poker Bot World Champion. http://www.bluffbot.com (cited: 15th June, 2011).

[34] Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terene Schauenberg, Duane Szafron. Approximating Game-Theoretic Optimal Strategies for Full-Scale Poker. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 4-8, 2003.

[35] Yoav Shoham, Rob Powers, Trond Grenager. Multi-Agent Reinforcement Learning: A Critical Survey. Technical report, Stanford University, 2003.

[36] Phil Gordon. *Phil Gordon's Little Green Book*. Simon Spotlight Entertainement, 2005.

[37] Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron. Poker as a Testbed for Machine Intelligence Research. In: *Advances in Artificial Intelligence (AI-98)*, pp. 228-238. Springer-Verlag, 1998.

[38] Michael Johanson, Michael Bowling. Data Biased Robust Counter Strategies. In: *Proceedings of the Twelfh International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 264-271, 2009.

[39] Phil Hellmuth. *Play Poker Like the Pros*. Collins Livig, 2003.

[40] Andrew Gilpin, Tuomas Sandholm, Troels Sørensen. Potential-aware Automated Abstraction of Sequential Games, and Holistic Equilibrium Analysis of Texas Hold'em Poker. In: *Proceedings of the 22nd National Conference on Artificial Intelligence*, vol. 1, pp. 50-57, 2007.

[41] João Ferreira. *Opponent Modeling in Texas Hold'em: Learning Pre-Flop Strategies in Multiplayer Tables*. Ms. C. thesis, Faculty of Engineering of the University of Porto, 2008.

[42] Dinis Félix. *Artificial Intelligence Techniques in Games with Incomplete Information*. Ms. C. thesis, Faculty of Engineering of the University of Porto, 2008.

[43] PokerSource: poker hand evaluator and more. http://pokersource.sourceforge.net (cited: 16th June, 2011).

[44] David Sklansky. *Tournament Poker for Advanced Players*. Two Plus Two Publishing, 2002.

[45] Poker-Academy Forums. Meerkat API and AI Discussion. http://forums.poker-academy.com/viewforum.php?f=3 (cited: 16th June 2011).

[46] Rickard Anderson. *Pseudo-Optimal Strategies in No-Limit Poker*. Ms. C. thesis, Umea University, 2006.

[47] Poker-Academy. Poker Academy – Our Poker Community, 2010. http://poker-academy.com/community.php (cited: 16th June, 2011).

[48] Annual Computer Poker Competition. http://www.computerpokercompetition.org/ (cited: 16th June, 2011).

[49] Open Meerkat Poker Testbed. http://code.google.com/p/opentestbed/ (cited: 17th June, 2011).

[50] Darse Billings, Lourdes Peña, Jonathan Schaeffer, Duane Szafron. Using Probabilistic Knowledge and Simulation to Play Poker. In: *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference (AAAI'99/IAAI'99)*, pp. 697-703. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1999.

[51] Darse Billings, Denis Papp, Jonathan Schaeffer, Duane Szafron. Opponent Modeling in Poker. In: Proceedings of the 15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI'98/IAAI'98), pp. 493-499. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1998.

[52] Darse Billings, Aaron Davidson, Jonathan Schaeffer, Duane Szafron. The Challenge of Poker. In: Artificial Intelligence, vol. 134, no. 1-2, pp. 201-240. Elsevier Science Publishers Ltd, Essex, UK, 2002.

[53] Daphne Koller, Nimrod Megiddo, Bernhard von Stengel. Fast Algorithms for Finding Randomized Strategies in Game Trees. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC'94), pp. 750-759. ACM, New York, NY, USA, 1994.

[54] Michael Littman, Martin Zinkevih. The 2006 AAAI Computer Poker Competition. In: ICGA Journal 29, pp. 166-167, 2006.

[55] Bluffbot – Poker AI, November, 2007. http://pokerbots.org/wiki/index.php/Bluffbot (cited: 17th June, 2011).

[56] 2007 AAAI Computer Poker Competition. http://webdocs.cs.ualberta.ca/~pokert/2007/index.php (cited: 17th June, 2011).

[57] Andrew Gilpin, Tuomas Sandholm. Better Automated Abstraction Techniques for Imperfect Games, with Application to Texas Hold'em. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), pp. 1168-1175, 2007.

[58] Samid Hoda, Andrew Gilpin, Javier Peña. A Gradient-Based Approach for Computing Nash Equilibria of Large Sequential Games. Unpublished manuscript. Available at http://andrew.cmu.edu/user/shoda/1719.pdf, 2007.

[59] World Poker Robot Championship, 2010. http://www.poker-academy.com/wprc/ (cited: 17th June, 2011).

[60] Man vs Machine – Results. http://www.poker-academy.com/man-machine/results.php (cited: 17th June, 2011).

[61] Man vs Machine 2008 – Results. http://www.poker-academy.com/man-machine/2008/results.php (cited: 17th June, 2011).

[62] Michael Bowling, Michael Johanson, Neil Burch, Duane Szafron. Strategy Evaluation in Extensive Games with Importance Smapling. In: Proceedings of the 25th International Conference on Machine Learning (ICML'08), pp. 72-29, 2008.

[63] Steve Brecher. brecware Software. http://www.stevebrecher.com/Software/software.html (cited: 17th June. 2011).

[64] Dinis Félix, Luís Paulo Reis. Opponent Modelling in Texas Hold'em Poker as the Key to Success. In *Proceeding of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, Nikos Avouris (Eds.), pp. 893-894. IOS Press, Amsterdam, The Netherlands, 2008.

Appendix A – Glossary of Poker Terms

- **All-in**. To have one's entire stake committed to the current pot. Action continues toward a side pot, with the all-in player being eligible to win only the main pot.
- **All-in Equity**. The expected value income of a hand assuming the game will proceed to the showdown with no further betting (i.e., a fraction of the current pot, based on all possible future outcomes).
- **Bad Beat**. An unlucky loss. In particular, losing a game where the opponent probably should have folded, but instead got extremely lucky to win.
- **Bet**. To make the first wager of a betting round (compare raise).
- **Bet for Value**. To bet with the expectation of winning if called (compare bluff).
- **Big Bet**. The largest bet size in Limit poker (e.g., $20 in $10-$20 Hold'em).
- **Big Blind (sometimes called the Large Blind)**. A forced bet made before the deal of the cards (e.g., $10 in $10-$20 Hold'em, posted by the second player to the left of the button).
- **Blind**. A forced bet made before the deal of the cards (see small blind and big blind).
- **Bluff**. To play a weak hand as though it were strong, with the expectation of losing if called (see also semi-bluff and pure bluff, compare bet for value).
- **Board (or Board Cards)**. The community cards shared by all players.
- **Board Texture**. Classification of the type of board, such as having lots of high cards, or not having many draws (see dry).
- **Button**. The last player to act in each betting round in Texas Hold'em. Also called the dealer button, representing the person who would be the dealer in a home game.
- **Call**. To match the current level of betting. If the current level of betting is zero, the term check is preferred.
- **Cap**. (a) The maximum number of raises permitted in any single round of betting (typically four in Limit Hold'em, but occasionally unlimited). (b) (vt) To make the

last permitted raise in the current betting round (e.g., after a bet, raise, and re-raise, a player caps the betting).

- **Check**. To decline to make the first wager of a betting round (compare call).
- **Check-Raise**. To check on the first action, with the intention of raising in the same betting round after an opponent bets.
- **Community Cards**. The public cards shared by all players.
- **Connectors**. Two cards differing by one in rank, such as 7-6. More likely to make a straight than other combinations.
- **Dominated**. A Hold'em hand that has a greatly reduced chance of winning against another because one or both cards cannot make a useful pair (e.g., KQ is dominated by AK, AQ, AA, KK, and QQ, but not by AJ or JJ).
- **Draw**. A holding with high potential to make a strong hand, such as a straight draw or a flush draw (compare made hand).
- **Draw Potential**. The relative likelihood of a hand improving to be the best if it is currently behind.
- **Drawing Dead**. Playing a draw to a hand that will only lose, such as drawing to a flush when the opponent already holds a full house.
- **Drawing Hand**. A hand that has a good draw (compare made hand).
- **Dry**. Lacking possible draws or betting action, as in a dry board or a dry game.
- **Equity (or Pot Equity)**. An estimate of the expected value income from a hand that accounts for future chance outcomes, and may or may not account for the effects of future betting (e.g., all-in equity).
- **Expected Value (EV) (also called mathematical expectation)**. The average amount one expects to win in a given game situation, based on the payoffs for each possible random outcome.
- **Flop**. The first three community cards dealt in Hold'em, followed by the second betting round (compare board).
- **Fold**. To discard a hand instead of matching the outstanding bet, thereby losing any chance of winning the pot.
- **Fold Equity**. The equity gained by a player when an opponent folds. In particular, the positive equity gained despite the fact that the opponent's fold was entirely correct.

- **Forward Blinds**. The logical extension of blinds for heads-up (two-player) games, where the first player posts the small blind and the second player (button) posts the big blind (compare reverse blinds). (Both rules are seen in practice, with various casinos and online card rooms having different policies for multi-player games that have only two active players).
- **Free-Card Danger**. The risk associated with allowing an opponent to improve and win the pot without having to call a bet (in particular, when they would have folded).
- **Free-Card Raise**. To raise on the flop intending to check on the turn.
- **Game**. (a) A competitive activity in which players contend with each other according to a set of rules (in poker, a contest with two or more players). (b) A single instance of such an activity (in poker, from the initial dealing of the cards to the showdown, or until one player wins uncontested).
- **Game Theory**. Among serious poker players, game theory normally pertains to the optimal calling frequency (in response to a possible bluff), or the optimal bluffing frequency. Both depend only on the size of the bet in relation to the size of the pot.
- **Hand**. (a) A player's private cards (e.g., two hole cards in Hold'em). (b) One complete game of poker (see game (b)).
- **Heads-up**. A two-player (head-to-head) poker game.
- **Hole Card**. A private card in poker (Texas Hold'em, Omaha, 7-Stud, etc.).
- **Implied Odds**. (a) The pot odds based on the probable future size of the pot instead of the current size of the pot (positive or negative adjustments). (b) The extra money a strong hand stands to win in future betting rounds (compare reverse implied odds).
- **Kicker**. A side card, often deciding the winner when two hands are otherwise tied (e.g., a player holding Q-J when the board is Q-7-4 has top pair with a Jack kicker).
- **Large Blind (usually called the Big Blind)**. A forced bet made before the deal of the cards (e.g., $10 in $10-$20 Hold'em, posted by the second player to the left of the button).
- **Loose Game**. A game having several loose players.
- **Loose Player**. A player who does not fold often (e.g., one who plays most hands at least to the op in Hold'em).

- **Made Hand**. A hand with a good chance of currently being the best, such as top pair on the op in Hold'em (compare draw).
- **Mixed Strategy**. Handling a particular type of situation in more than one way, such as to sometimes call, and sometimes raise.
- **Offsuit**. Two cards of different suits (also called unsuited, compare suited).
- **Open-Ended Draw**. A draw to a straight with eight cards to make the straight, such as 6-5 with a board of Q-7-4 in Hold'em.
- **Outs**. Cards that will improve a hand to a probable winner (compare draw).
- **Pocket Pair**. Two cards of the same rank, such as 6-6. More likely to make three of a kind than other combinations (see set).
- **Post-flop**. The actions after the flop in Texas Hold'em, including the turn and river cards interleaved with the three betting rounds, and ending with the showdown.
- **Pot**. The common pool of all collected wagers during a game.
- **Pot Equity (or simply Equity)**. An estimate of the expected value income from a hand that accounts for future chance outcomes, and may or may not account for the effects of future betting (e.g., all-in equity).
- **Pot Odds**. The ratio of the size of the pot to the size of the outstanding bet, used to determine if a draw will have a positive expected value.
- **Pre-fop**. The first round of betting in Texas Hold'em before the flop, beginning with the posting of the blinds and the dealing of the private hole cards.
- **Pure bluff**. A bluff with a hand that can only win if the opponent folds (compare semi bluff).
- **Pure Drawing Hand**. A weak hand that can only win by completing a draw, or by a successful bluff.
- **Raise**. To increase the current level of betting. If the current level of betting is zero, the term bet is preferred.
- **Raising for a Free-card**. To raise on the op intending to check on the turn.
- **Rake**. A portion of the pot withheld by the casino or host of a poker game, typically a percentage of the pot up to some maximum, such as 5% up to $3.
- **Re-raise**. To increase to the third level of betting after a bet and a raise.
- **Reverse Blinds**. A special rule sometimes used for heads-up (two-player) games, where the second player (button) posts the small blind and the first player posts the big blind (compare forward blinds). (Both rules are seen in practice, with various

casinos and online card rooms having different policies for multi-player games that have only two active players).

- **Reverse Implied Odds**. The unaccounted (negative) money a mediocre hand stands to lose in future betting rounds (compare implied odds (b)).
- **River**. The fifth community card dealt in Hold'em, followed by the fourth (and final) betting round.
- **Semi-bluff**. A bluff when there are still cards to be dealt, with a hand that might be the best, or that has a reasonable chance of improving to the best if it is called (compare pure bluff).
- **Second pair**. Matching the second highest community card in Hold'em, such as having 7-6 with a board of Q-7-4.
- **Session**. A series of games, typically lasting several hours in length.
- **Set**. Three of a kind, formed with a pocket pair and one card of matching rank on the board. A very powerful and well-disguised hand (compare trips).
- **Short-handed Game**. A game with less than the full complement of players, such as a Texas Hold'em game with five or fewer players.
- **Showdown**. The revealing of cards at the end of a game to determine the winner.
- **Side pot**. A second pot for the remaining active players after another player is all-in.
- **Slow-play**. To check or call a strong hand as though it were weak, with the intention of raising in a later betting round (compare smooth-call and check raise).
- **Small Bet**. The smallest bet size in Limit poker (e.g., $10 in $10-$20 Hold'em).
- **Small Blind**. A forced bet made before the deal of the cards (e.g., $5 in $10-$20 Hold'em, posted by the first player to the left of the button).
- **Smooth-call**. To only call a bet instead of raising with a strong hand, for purposes of deception (as in a slow-play).
- **Suited**. Two cards of the same suit, such as both Hearts. More likely to make a flush than other combinations (compare off suit or unsuited).
- **Table Image**. The general perception other players have of one's play.
- **Table Stakes**. A poker rule allowing a player who cannot match the outstanding bet to go all-in with his remaining money, and proceed to the showdown (also see side pot).

- **Texture of the Board**. Classification of the type of board, such as having lots of high cards, or not having many draws (see dry).
- **Tight Player**. A player who usually folds unless the situation is clearly profitable (e.g., one who folds most hands before the flop in Hold'em).
- **Time Charge**. A fee charged to the players in a poker game by a casino or other host of the game, typically collected once every 30 minutes.
- **Top Pair**. Matching the highest community card in Hold'em, such as having Q-J with a board of Q-7-4.
- **Trap**. To play a strong hand as though it were weak, hoping to lure a weaker hand into betting. Usually a check-raise or a slow-play.
- **Trips**. Three of a kind, formed with one hole card and two cards of matching rank on the board. A strong hand, but not well-disguised (compare set).
- **Turn**. The fourth community card dealt in Hold'em, followed by the third betting round.
- **Unsuited**. Two cards of different suits (also called off suit, compare suited).
- **Value Bet**. To bet with the expectation of winning if called (compare bluff).
- **Wild Game**. A game with a lot of raising and re-raising. Also called an action game.