HoldemML: A Framework to generate No Limit Hold'em Poker Agents from Human Player Strategies

Luís Filipe Teófilo
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Portugal
Laboratório de Inteligência Artificial e Ciência de Computadores, Universidade do Porto, Portugal
Porto, Portugal
luis.teofilo@fe.up.pt

Luís Paulo Reis
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Portugal
Laboratório de Inteligência Artificial e Ciência de Computadores, Universidade do Porto, Portugal
Porto, Portugal
lpreis@fe.up.pt

Abstract—**Developing computer programs that play Poker at human level is considered to be challenge to the A.I. research community, due to its incomplete information and stochastic nature. Due to these characteristics of the game, a competitive agent must manage luck and use opponent modeling to be successful at short term and therefore be profitable. In this paper we propose the creation of No Limit Hold'em Poker agents by copying strategies of the best human players, by analyzing past games between them. To accomplish this goal, first we determine the best players on a set of game logs by determining which ones have higher winning expectation. Next, we define a classification problem to represent the player strategy, by associating a game state with the performed action. To validate and test the defined player model, the HoldemML framework was created. This framework generates agents by classifying the data present on the game logs with the goal to copy the best human player tactics. The created agents approximately follow the tactics from the counterpart human player, thus validating the defined player model. However, this approach proved to be insufficient to create a competitive agent, since the generated strategies were static, which means that they are easy prey to opponents that can perform opponent modeling. This issue can be solved by combining multiple tactics from different players. This way, the agent switches the tactic from time to time, using a simple heuristic, in order to confuse the opponent modeling mechanisms.**

Poker; Data Mining; Machine Learning; Opponent Modeling; Artificial Intelligence (Keywords)

I. INTRODUCTION

Poker is a game that is increasingly becoming a field of interest for the AI research community on the last decade. The way to develop agents for Poker is rather different than developing for games like chess or checkers – complete information games. In these games, most methods are based on decision trees (e.g. minimax) in combination with heuristics to score each decision. Notable results in complete information games were achieved in the past. For instance, one of the most famous cases of success was the computer Deep Blue [1], which was the first A.I. to beat a Chess champion in a series of games.

Unlike Chess, Poker is an incomplete information game, because each player can only see its cards or the community cards. For this reason, the decision trees in Poker are probabilistic, which means that each branch of the tree has a probability of occurring, considering the type of opponent. Therefore, a competitive agent must model the opponents, to identify the probability of each possible action. By identifying the opponents playing style, it is possible to predict their possible actions and therefore make a decision that has better probability of success [2, 3].

The main goal of this work is to determine on how strategies used in the past by good human players can be used to support the creation of Poker agents. This work is distributed in the subsequent steps:

- Extract a good amount of games between good human players;
- Determine the best players present on the database;
- Define a classification problem: game state variables that can influence player's decision and the possible outcomes;
- Create a strategy by classifying game state instances;
- Create a framework that facilitates the replication of the above steps;
- Create and test generated agents.

The rest of the paper is structured as follows. Section II briefly describes Texas Hold'em Poker. Section III describes related work about approaches followed earlier to create Poker agents. Section IV describes the characteristics of the dataset that was used in this work. Section V presents the classification problem and the comparison of the machine learning algorithms that were used to solve it. Section VI describes the architecture of the HoldemML framework that was created to generate agents from game logs as well as some details about the implementation of the agent. Section VII describes tests and results of the games played by the produced agents. Finally, the section VIII presents the paper main conclusions and some pointers for future work.

II. TEXAS HOLD'EM POKER

Poker is a generic name for literally hundreds of games, but they all fall within a few interrelated types [4]. It is a card game

in which players bet that their hand is stronger than the hands of their opponents. All bets go into the pot and at the end of the game, the player with the best hand wins. Other way of winning is forcing all opponents to forfeit the hand, by raising the current bet. This work is focused on No Limit Texas Hold'em Poker, the most popular Poker variant nowadays.

A. Hand Ranking

A poker hand is a set of five cards that identifies the strength of a player in a game of poker. The hand is composed by the player pocket cards and the community cards – cards that belong to all players. Joining the pocket and community cards, if there are more than 5 cards, the hand rank is the best possible rank of all 5 card combinations.

The possible hand ranks are (stronger ranks first): Royal Flush (top sequence of same suit), Straight Flush (sequence of same suit), Four of a Kind (4 cards with same rank), Full House (Three of a Kind + Pair), Straight (sequence of cards), Three of a Kind (3 cards of the same rank), Two Pair, One Pair (2 cards with same rank) and Highest Card (when does not qualify to any other ranks).

B. No Limit Texas Hold'em

No Limit Texas Hold'em is a Poker variation that uses community cards. At the beginning of every game, two cards are dealt for each player. A dealer player is assigned and marked with a dealer button. The dealer position rotates clockwise from game to game. After that, the two players to the left of dealer post the blind bets. The first player is called small blind, and the next one is called big blind. They respectively post half of minimum and the minimum bets. The player that starts the game is the one on the left of the big blind.

After configuring the table, the game begins. The game is composed by four rounds of betting (Pre-Flop, Flop, Turn, River). In each round, in its turn, the player can perform the following actions: Bet, Call, Raise, Check, Fold or All-In.

In any game round, the last standing player wins the game and therefore the pot. If the River round reaches the end, the winner is the one with the highest ranked hand.

This variation of Poker is No Limit, which means that at any round of the game everyone is allowed to bet any amount above the minimum bet.

III. Related Work

Most noticeable achievements in Computer Poker are from the Computer Poker Research Group (CPRG) [5] from University of Alberta, mostly in the variant Limit Texas Hold'em. One of the most renowned publication of the group is [6] where is described a complete analysis of the evolution of artificial poker agent architectures, demonstrating strengths and weaknesses of each architecture both in theory and in practice. Another significant publication is [7] where a perfect strategy was defined for a very simple variant of Poker (Rhode Island). Another near-perfect strategy was achieved in [8] for Heads-Up No Limit variant, using the Nash Equilibrium Theory. More recently, it should be emphasized the article [9], which describes a new and more effective technique for building counter strategies based on Restricted Nash Response. Some achievements were also made using data mining classifiers like in [10] where were studied evolutionary methods to classify in game actions.

Despite all the breakthroughs achieved by known research groups and individuals, no artificial poker playing agent is presently known capable of beating the best human players.

A. Opponent Modeling in Poker

A large number of opponent modeling techniques are based on real professional poker players' strategies, such as David Sklansky, who published one of the most renowned books on poker strategy [4].

One possible classification is tightness and aggressiveness of the players. A player is tight if he plays 28% or less hands, and loose if he plays more than 28% of the times. With regard to aggressiveness, the player is aggressive if he has an aggression factor (Equation 1) over 1.0; otherwise it is a passive player.

$$\text{AgressionFactor} = \frac{\text{Number of Raises}}{\text{Number of Calls}} \quad (1)$$

B. Poker Hand Rank Evaluation

Hand rank evaluation consists in checking if a given Poker hand is better than another. Usually, the Hand Rank Evaluator takes a poker hand and maps it to a unique integer which is the score of the hand. Hands with equivalent rank have the same score and ands with lesser rank have a lower score. For poker AI, it is absolutely critical to have the fastest hand evaluators possible [11]. Any poker agent may have to evaluate thousands of hands for each action it will take. The fastest known evaluator is *TwoPlusTwo Evaluator* [12], which can evaluate about 15 millions of hands per second.

C. Odds Calculation

Evaluating the rank of the hand is giving a score to a set of cards. A Poker AI normally uses hand rank formulas to determine the hand odds. Hand odds measure the strength or the potential strength of a hand. This can be done by comparing the hand with the possible opponents' hands. This prediction is used to help measuring the risk of an action.

There are various known ways to determine the odds of a hand:

- Chen Formula [13]: this formula can determine the relative value of a 2 card hand.
- Hand Strength [6, 11, 14]: determines how many hands are ahead of ours, only taking into account the current available community cards.
- Hand Potential [6, 11, 14]: The hand potential is an algorithm that calculates PPOT and NPOT. The PPOT is the chance that a hand that is not currently the best improves to win at the showdown. The NPOT is the chance that a currently leading hand ends up losing. Therefore, they are used to estimate the flow of the game.

- Effective Hand Strength [6, 11, 14]: Combines the hand strength and the hand potential formulas in an unique measure.

D. Agent development tools

There are some software tools that can aid the development of a Poker agent. Most notable is the Meerkat API [11] that easily allows the creation of Poker agents. The Meerkat agents can be tested using game simulators that support this API. The original simulator is Poker Academy [15], but there are other open source solutions like Open Meerkat Test Bed [16].

IV. POKER DATA EXTRACTION AND ANALYSIS

To create player models from past human games we required a great amount of Poker games to analyze. The selected data source is composed of game logs from online casinos. The Poker game logs represent the list of actions of the players during the games on a given Poker table. These files don't represent the state of the game. So, to analyze the game it is necessary to replay the game with the same actions and card distribution. For each action, the game state and the action is stored in a database.

Obtaining information from these documents is difficult since these files typically do not contain an organized structure, making it difficult to parse the information. Moreover, there is no standard to represent game movements: each online casino has its own representation of logs. For this reason, to combine data from multiple sources, a new parser is needed for each game log format. The logs were converted to a common format that is presented in section VI.

The package of game logs that was used in this work can be found here [17]. Some characteristics of the game logs can be found on table 1.

TABLE I. GAME LOGS CHARACTERISTICS

Characteristic	Value
Number of games	51.377.820
Number of players	158.035
Number of showdowns	2.323.538
Number of players with 500 or more showdowns	183

To characterize a game state, from the standpoint of the player, in order to learn his tactic, it is necessary to know which cards the player had. As it can be seen on table 1, the percentage of games in which card exposal occurs is very reduced, and only 183 players showed their cards more than 500 times. These were the players that were selected for this work.

After obtaining the game data, we selected the best players available in order to learn the best tactics. For this task, a player list was generated containing some information about each player. The criteria used to choose a good player was its earnings as, for instance, a player with high negative earnings probably doesn't have a good tactic.

TABLE II. CHARACTERISTICS OF THE EXTRACTED PLAYERS

Name	Game Count	Number of shows	Earnings	Winning expectation
John	15.763	638	1.003$	+ 0.06$
Kevin	20.660	838	30$	+ 0.00$
David	77.598	2.103	14.142$	+ 0.18$
Jeff	33.257	882	-4.945$	- 0.15$

The table II presents the players that were analyzed on this paper. Players with different expectations (average money per game) were chosen, to conclude if players that won more money on the past generate agents with better game performance. It should be noted that a player with negative earnings was included (Jeff). This served for testing purposes, to check if a tactic generated from a theoretical bad player loses against a tactic from a theoretical good player.

V. LEARNING POKER STRATEGIES

To learn the tactics of the players, we used supervised learning algorithms. To facilitate both the learning process and the implementation of the agent, WEKA [18] software was used. For each player it was obtained an ARFF file with all game states, of the chosen players, in games with showdown. The characteristics used to classify the players' tactics were the ones that might influence the players' decisions during the game: position in table, money, last opponent's action, Sklansky classification of last opponent.

```
@relation poker
@attribute positionScore numeric
@attribute effectiveHandStrength numeric
@attribute percentageMoneyToBet numeric
@attribute percentageMoneyOnTable numeric
@attribute possibleWins numeric
@attribute lastOpponentAction {call,raise}
@attribute isLastPlyrAggres {true, false}
@attribute isLastPlyrTight {true, false}
@attribute action {call, raise5, raise10,
raise15, raise20…}
```

The position score attribute represents the value of the position of the player on the table. The value is greater if the player is closer to the dealer. When the player is the dealer, the position score value is 1. The effective hand strength [6, 11, 14] attribute characterizes the odds of the player winning the round with the current cards. The percentage of money to bet represents the percentage of the player's chip amount that he has to put on the table to call the hand. The percentage of money on table defines the percentage of chips that the player has already put on the table, in the current game. The possible wins attribute represents how much chips the player can win comparatively to his own stack. Next, we have the last opponent action (call or raise) and the Sklansky classification

of the last opponent: aggressive or passive (equation 1) and tight or loose [4].

The last attribute is nominal and it represents the player's action, being, for that reason, the class attribute. Another important fact to note is that each player has four ARFF files associated with it, each one representing a game round (Pre-Flop, Flop, Turn, River) . This is because tactics used during the game tend to be different in each game round, due to factors such as the varying number of community cards available.

Different classifiers were tested to build this model. The classifiers that obtained a smaller average error (using tenfold cross validation) were search trees, more particularly Random Forest Trees (Figure 1). The errors were relative high as was expected as players tend to change tactic during the game, making it difficult to find a consistent pattern.

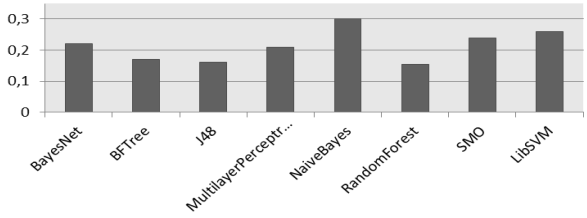

Figure 1. Average classifier error rate.

The classifiers error rate was also analyzed per round (Figure 2). It can be observed that the error is much higher in Flop and Turn rounds than in Pre Flop and River rounds. This was expected for Pre Flop round, since there are no communities cards, the tactics tend to be simpler. As for River round, this can be explained by the lower number of players that reach this round.

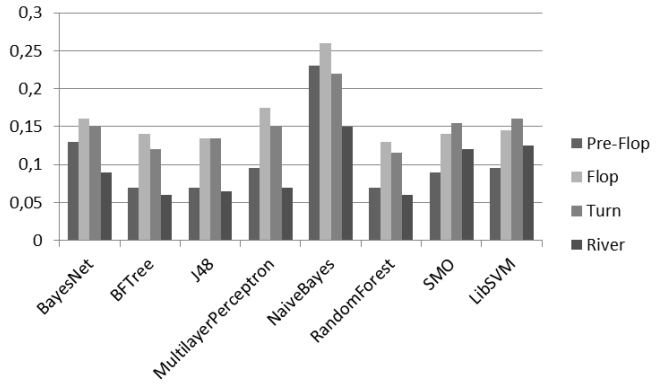

Figure 2. Average classifier error per round

VI. HOLDEMML FRAMEWORK

The **HoldemML Framework** global architecture can be seen in Figure 3. Initially we have different data sources that contain poker game data. Since each data source can represent the data in a different way, we must convert the data into a common format in order to combine the information from different logs, because the same player might play in different online casinos.

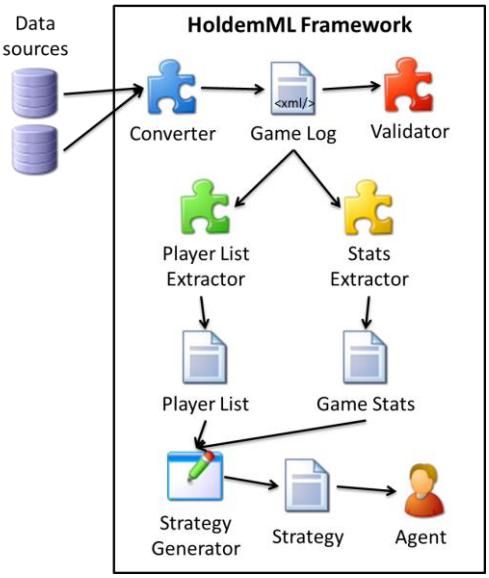

Figure 3. HoldemML Framework

The role of converting data into a common format is played by **HoldemML Converter**. The output files are in XML format and follow the following tree scheme (Figure 4).

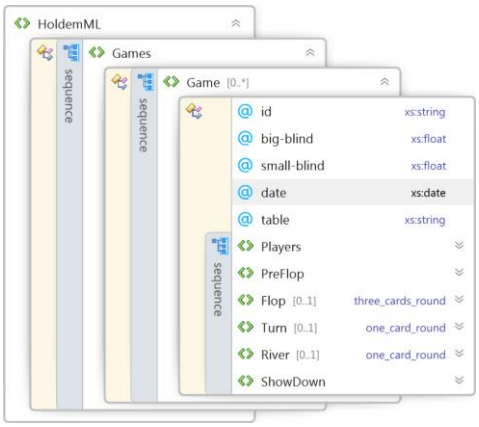

Figure 4. HoldemML XML Schema

It is also possible to verify whether the documents from data sources are already in the format set as default, using the **HoldemML Validator** module.

Having the data ready to process, the next step was to extract the game variables that define the game state and the action performed by the players to use as training set for the classifier. The module that extracts the game variables is **HoldemML Stats Extractor**, generating a game stats file. **HoldemML Stats Extractor** requires as input the games logs and a player list. The player list is created by **HoldemML Player List Extractor** and it contains the list of all players present on the set of game logs as well as information about the player performance in the games that it participated. The player list is also used to determine players with little game

participation which, for that reason, should not be used to train the classifiers.

After generating the game stats file and the player list file, they are used by **HoldemML Strategy Generator** to create a strategy file. The game stats file is the main source of information to generate a strategy, but the player list is also useful to give information about opponents.

After generating the player models, it is possible to use the **HoldemML Strategy Generator** to create an agent, by choosing the tactics by which the strategy of the agent to be generated is composed and the heuristic that changes the tactic through the game. Three simple heuristics were included: random tactic change; time to time tactic change; change tactic when the agent is losing money with the current one.

The generated agents use the Meerkat API [11]. The agent's strategy is described in Figure 5. The agent contains various tactics and a tactic chooser module. The tactic chooser module recovers information about the events that occur on the game table and when is called for action it chooses one of the tactics to determine the action.

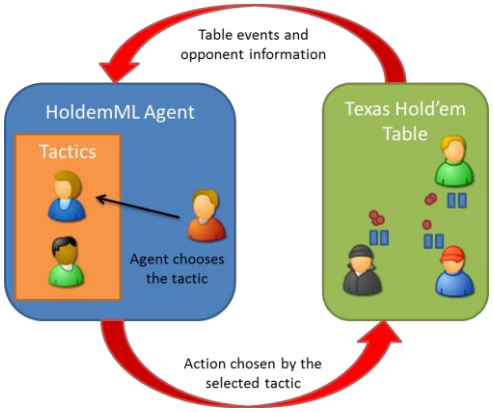

Figure 5. HoldemML agent behaviour

One potential problem with this player model is that the agent never folds its hands, since these actions weren't considered on the training set because the players don't show their cards when they fold. To solve this problem, a criterion was defined to fold some hands. The defined criterion was that if the hand strength [6] is below 50%, than the agent has a probability of folding equal to its tightness level. The agent also folds when it can't find any class with high significance level.

VII. Tests and Results

After the implementation of the framework, four types of tests were used to validate this approach: behavior tests, tests between generated tactics, tests against other agents previously developed and strategy tests. All tests were made using the Open Meerkat Test Bed, with 1.000 cash games per test and table seat permutation. The players used on the test are described on table II. The behavior tests are presented on table III. As it can be observed, the agent approximately conserved the real player Sklansky's classification.

TABLE III. Agent Behaviour Testing

Name	Agent AF	Real player AF	Agent tightness	Real player tightness	Agent classif.	Real player classif.
John	4,77	5,12	0,32	0,31	L.Agg.	L.Agg.
Kevin	1,55	1,49	0,25	0,23	T.Agg.	T.Agg.
David	16,04	13,22	0,19	0,19	T.Agg.	T.Agg.
Jeff	9,42	8,40	0,44	0,33	L.Agg.	L.Agg.

In Figure 6 it is presented the bankroll evolution in a series of games between HoldemML agents. The humans that won more money in real life generated a better agent.

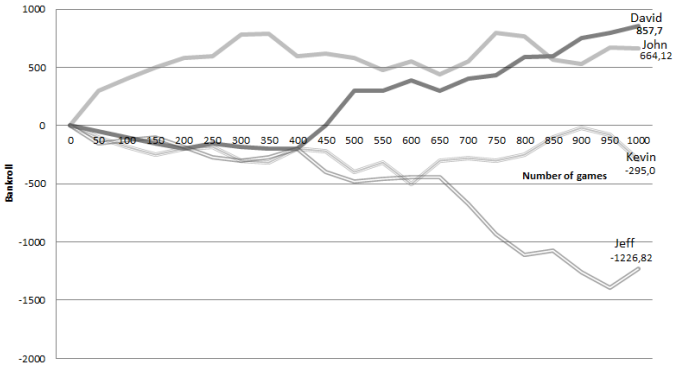

Figure 6. Games between HoldemML agents

Next, we tested HoldemML John agent against HSB Bot (Figure 7) which is an agent that chooses his actions based on its effective hand strength [6]. HoldemML John clearly won the match by a large margin of 1.677, 30$. HoldemML John was also tested against an agent that always calls its hands, achieving similar results.

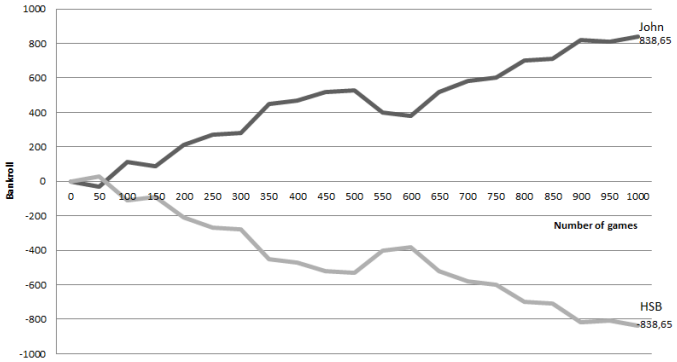

Figure 7. Games between HoldemML John and Hand Strength Bot

Finally, we tested HoldemML John agent against MCTS Bot [19]. The HoldemML agent was totally defeated (Figure 8), because the MCTS Bot is capable of opponent modeling.

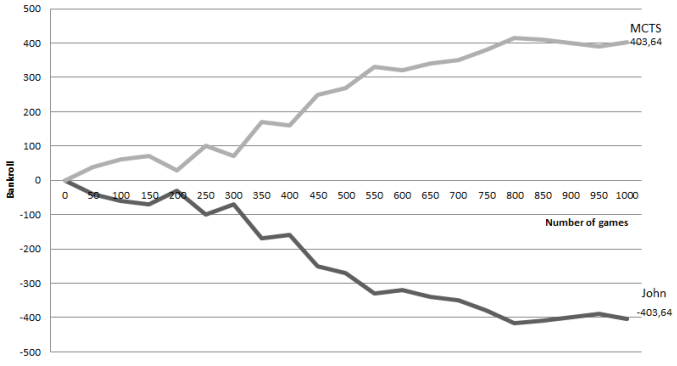

Figure 8. Games between HoldemML John and MCTS Bot[19]

The results improved a lot by using a strategy that combines multiple tactics of generated agents (Figure 9). The tactic changes when the agent is losing money, therefore it confuses the opponent modeling mechanisms of MCTS Bot. We can observe some cycles on the chart due to this fact. When the agent starts losing money, it changes its tactic, and after a little while it starts winning again, until the moment that the MCTS Bot discovers the new tactic.

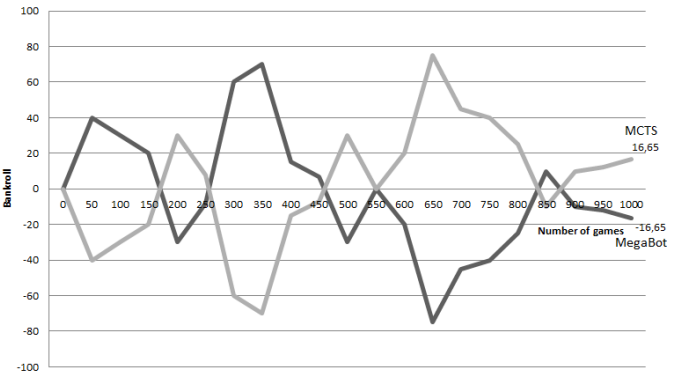

Figure 9. Games between HoldemML MegaBot and MCTS Bot [19]

VIII. Conclusions and Future Work

There is still a long way to go to create an agent that plays poker at the level of the best human players. This research presented an approach based on supervised learning methodologies, where the agent relies on pre learned tactics based on past human experience to decide upon its actions. The created agents are not competitive against good poker players that are capable of opponent modeling. However the results greatly improved after combining various tactics, which means that an agent should not use a static tactic in a game like Poker.

This approach can be promising for good human players, since they can create an agent based on their logs that will play like for them, autonomously. Plus, since these players have access to their own logs, they have much more information to create more representative player models.

The generated agents could be improved in the future, by defining a more complex player model or by specifying better heuristics to change tactic along the game.

Acknowledgements

I would like to thank to Luís Paulo Reis for providing useful information to complete this article and for making the final revision. I also would like to thank to Fundação para a Ciência e a Tecnologia for supporting this work by providing my PhD Scholarship SFRH/BD/71598/2010. Finally, I thank to Departamento de Engenharia Informática and ProDEI for providing financial support for the conference registration.

References

[1] Feng-Hsiung Hsu, Behind Deep Blue; Building the Computer that Defeated the World Chess Champion. Princeton University Press (2002)

[2] Darse Billings, Dennis Papp, Jonathan Schaeffer J, Duane Szafron; Opponent modeling in poker. In proceedings of AAAI '98/IAAI '98, Madison, Wisconsin, United States, 1998. pp 493-499

[3] Aaron Davidson; Opponent modeling in poker. M.Sc., University of Alberta, Edmonton, Alberta, Canada (2002)

[4] David Sklansky; The Theory of Poker: A Professional Poker Player Teaches You How to Think Like One. Two Plus Two (2002)

[5] Computer Poker Research Group (CPRG) Homepage http://webdocs.cs.ualberta.ca/~games/poker/. Accessed 10 January 2011

[6] Darse Billings Algorithms and Assessment in Computer Poker. Ph.D., University of Alberta, Edmonton, Alberta, Canada (2006)

[7] Andrew Gilpin, Tuomas Sandholm; Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In proceedings of 6th international joint conference on Autonomous agents and multiagent systems, Honolulu, Hawaii, 2007. ACM, pp 1-8

[8] Peter Bro Miltersen, Troels Bjerre Sørensen; A near-optimal strategy for a heads-up no-limit Texas Hold'em poker tournament. In: 6th international joint conference on Autonomous agents and multiagent systems, Honolulu, Hawaii, 2007. ACM, pp 1-8

[9] Michael Johanson, Michael Bowling; Data Biased Robust Counter Strategies. In proceedings of AISTATS, 2009. pp 264-271

[10] Brien Beattie, Garrett Nicolai, David Gerhard, Robert J. Hilderman; Pattern Classification in No-Limit Poker: A Head-Start Evolutionary Approach. In proceedings of 20th conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, Montreal, Quebec, Canada, 2007. Springer-Verlag, pp 204-215

[11] Dinix Felix, Luís Paulo Reis; An Experimental Approach to Online Opponent Modeling in Texas Hold'em Poker. In proceedings 19th SBIA: Advances in Artificial Intelligence, Savador, Brazil, 2008. Springer-Verlag, pp 83-92

[12] Poker Bot Artificial Intelligence Resources. http://spaz.ca/poker/. Accessed 18 January 2011

[13] Bill Chen, Jerrod Ankenman; The Mathematics of Poker. Conjelco (2006)

[14] Dinix Félix, Luís Paulo Reis; Opponent Modelling in Texas Hold'em Poker as the Key for Success. In proceedings of 18th ECAI 2008, Amsterdam, Netherlands, 2008. IOS Press, pp 893-894

[15] Poker Academy. http://www.poker-academy.com/. Accessed 28 December 2010.

[16] Opentestbed. http://code.google.com/p/opentestbed/. Accessed 28 December 2010

[17] Out Flopped Poker Q & A – Obfuscated data mined hand histories. http://www.outflopped.com/questions/286/obfuscated-datamined-hand-histories. Accessed 18 January 2011

[18] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten; The WEKA data mining software: an update. SIGKDD Explor News, 2009, pp 10-18

[19] Guy Broeck, Kurt Driessen, Jan Ramon; Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In proceedings 1st Asian Conference on Machine Learning: Advances in Machine Learning, Nanjing, China, 2009. pp 367-381