Adapting Strategies to Opponent Models in Incomplete Information Games: A Reinforcement Learning Approach for Poker

Luís Filipe Teófilo[1,2], Nuno Passos[2],
Luís Paulo Reis[1,3], and Henrique Lopes Cardoso[1,2]

[1] LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal
[2] FEUP – Faculty of Engineering, University of Porto, DEI, Portugal
[3] EEUM – School of Engineering, University of Minho, DSI, Portugal
{luis.teofilo,ei08029,hlc}@fe.up.pt,
lpreis@dsi.uminho.pt

Abstract. Researching into the incomplete information games (IIG) field requires the development of strategies which focus on optimizing the decision making process, as there is no unequivocal best choice for a particular play. As such, this paper describes the development process and testing of an agent able to compete against human players on Poker – one of the most popular IIG. The used methodology combines pre-defined opponent models with a reinforcement learning approach. The decision-making algorithm creates a different strategy against each type of opponent by identifying the opponent's type and adjusting the rewards of the actions of the corresponding strategy. The opponent models are simple classifications used by Poker experts. Thus, each strategy is constantly adapted throughout the games, continuously improving the agent's performance. In light of this, two agents with the same structure but different rewarding conditions were developed and tested against other agents and each other. The test results indicated that after a training phase the developed strategy is capable of outperforming basic/intermediate playing strategies thus validating this approach.

Keywords: Incomplete Information Games, Opponent Modeling, Reinforcement Learning, Poker.

1 Introduction

One of the fields with large focus on AI research is games. Because games have a limited set of well-defined rules, studying them allows for easy testing of new approaches making it possible to accurately measure their degree of success. This is done by comparing results of many games played against programs based on other approaches or against human players, meaning that games have a well-defined metric for measuring the development progress [1]. It is then possible to determine with more accuracy whether if the solution is optimal to solve a given problem. Also, the fact that games have a recreational dimension and present an increasing importance for the entertainment industry today motivates further research on this domain.

Remarkable results were achieved in games research, such as the well-known Deep Blue Computer, which was the first computer to ever defeat a human chess champion [2]. However such success has not yet been achieved for incomplete information games. This is because that their game state is not fully visible which means that there are hidden variables/features. Therefore decision making in these games is more difficult, because predictions about the missing data must be made. This makes it almost impossible to obtain an optimal solution. Poker is a very popular game that presents these characteristics because players do not know the cards of their opponents.

The research on Computer Poker has been active in the past years. Several Poker playing agents were developed but none of them has reached a level similar to a expert human player. In order to overcome the limitations found in previously developed agents, a new agent has been developed. This approach tries to mimic human players by combining opponent models used by expert players and a reinforcement learning method. The usage of reinforcement learning in the conception of the agent's strategy allowed for good adaption of the agent to several pre-defined opponent types. Two different versions of this agent were implemented and they differ (only) in what regards the reward calculation.

The rest of the paper is organized as follows. Section 2 briefly shows this paper's background by presenting the game of Poker and basic opponent modeling. Section 3 describes related work, with particular emphasis on Computer Poker. Section 4 presents this paper's approach to create a Poker agent. Section 5 describes the validation process of this approach by indicating the experimental procedure and results. Finally, some conclusions are drawn and future research recommendations are as well suggested in section 6.

2 Background

Poker is a generic name for hundreds of games with similar rules [3], which are called variants. This work is focused on a simplified version of the Texas Hold'em variant, which is probably the most popular nowadays. Hold'em rules also have specific characteristics that allow for new developed methodologies to be adapted to other variants with reduced effort [4].

The game is based upon the concept of players betting that their current hand is stronger than the hands of their opponents. All bets throughout the game are placed in the pot and, at the end of the game, the player with the highest ranked hand wins. Alternatively, it is also possible to win the game by forcing the opponents to fold their hands by making bets that they are not willing to match. Thus, since the opponents' cards are hidden it is possible to win the game with a hand with lower score This is done by bluffing - convincing the opponents that one's hand is the highest ranked one.

2.1 Hand Ranking

A Poker hand is a set of five cards that defines the player's score. The hand rank at any stage of the game is the score given by the 5 card combination composed by player cards and community cards that has the best possible score. The possible hand ranks are (from stronger to weaker): Straight Flush (sequence of same suit), Four of a

Kind (4 cards with same rank), Full House (Three of a Kind + Pair), Straight (sequence), Three of a Kind (3 cards with same rank), Two Pair, One Pair (2 cards with same rank) and Highest Card (when not qualified to other ranks).

2.2 Rules of Simplified Texas Hold'em

The rules of the Poker variant used in this study represent a subset of the rules of Texas Hold'em, but only the initial round of the game is considered and it only allows for two players. In each game one of the players posts a mandatory minimum bet and the other one must bet half of that value. These players are called respectively big-blind and small-blind. After that, each player receives two cards – pocket cards – that are only seen by the player. The first player to act is the small-blind player and then the players play in turns. In each turn they can either match the highest bet (Call), increase that bet (Raise) or forfeit the game and lose the pot (Fold). When one of the players calls (except on the first big-blind player turn) five community cards are drawn from the deck and both players show their cards. The winning player is the one with the highest hand rank. If one of the players folds the other one wins the pot.

2.3 Opponent Modeling

One way of classifying opponents in this Poker variant is through VPIP and the aggression factor (AF) of the player [3]. The VPIP is the percentage of games in which the players raises at least one time. The aggression factor is the ration between the number of 'aggressive' actions and the number of 'passive' actions (equation 1).

$$AF = \frac{NumRaises}{NumCalls} \tag{1}$$

Poker experts classify opponents according to table 1, using the aforementioned indicators.

Table 1. Classifying Poker players using Agression Factor and VPIP

	$VPIP < 0,28$	$VPIP \geq 0,28$
$AF \geq 1$	TightAgressive	LooseAgressive
$AF < 1$	TightPassive	LoosePassive

3 Related Work

The first approach to build Poker agents was a rule-based approach which involves specifying the action that should be taken for a given game state [1]. The following approaches were based on simulation techniques [1, 5-7], i.e. generating random instances in order to obtain a statistical average and decide the action. These approaches led to the creation of agents that were able to defeat weak human opponents.

The great breakthrough in Poker research began with the use of Nash's equilibrium theory [8, 9]. Since then, several approaches based on Nash Equilibrium emerged: Best Response [10], Restricted Nash Response [1, 11] and data-biased response [12]. Currently, the best Poker agent Polaris [12] uses a mixture of these approaches.

Other recent methodologies were based on pattern matching [13, 14] and on the Monte Carlo Search Tree algorithm [14, 15].

A successful work closely related to this approach is [16]. It presents a reinforcement learning methodology to another simplified version of Poker – 1 card Poker. This approach uses Q-Learning to learn how to play against several opponent types.

Despite all the breakthroughs achieved, there is no known approach in which the agent has reached a level similar to a competent human player.

4 Proposed Approach

This section describes the structure of the two agents. These agents are similar in every aspect except reward conditions.

4.1 Common Structure

The agents were developed with a Q-Table containing the state-action pairs. The state ($\boldsymbol{\sigma}$) is defined as:

- **G**: A value representing a pair of cards that make the player's hand. This is useful since many hands have the same relative value (e.g. {2♣, 4♥} and {2♦, 4♣}).
- **P**: The player's seat on the table (big-blind or small-blind).
- **T**: A value representing the opponent type (Tight Aggressive, Tight Passive, Loose Aggressive and Loose Passive).
- **A**: A value representing the last action before the agent's turn (Call, Raise).

Each state has a direct correspondence to tuple (**C** – call weight, **R** – raise weight) as described in equation 2.

$$\begin{gathered} \sigma(G,P,T,A) \rightarrow (C,R) : C + R \leq 1; \\ G \in \{0 \ldots 127\}; P \in \{'Big,' Small'\}; T \in \{'TA',' TP',' LA',' LP'\}; \\ A \in \{'Call',' Raise'\};\ C,R \in [0,1] \end{gathered} \tag{2}$$

The Q-Table is initially empty and the weights are filled up with random numbers as there is need for them. The value of the weights stabilizes as the games proceed, so as to choose the option which maximizes profit. However convergence to stable weight values is not guaranteed because the game state to action mapping may not be sufficient to fully describe the defined opponent types.

When the agent plays, it searches the Q-Table to obtain the values of C and R so as to decide on the action to take. After retrieving these values, a random number ($N \in [0,1]$) is generated. The probability of choosing an action is as on equation 3:

$$Action = \begin{cases} Call, N \in [0, C] \\ Raise, N \in]C, C + R] \\ Fold\ otherwise \end{cases} \tag{3}$$

The flowchart present on figure 1 describes the complete process of update and usage of the Q-Table.

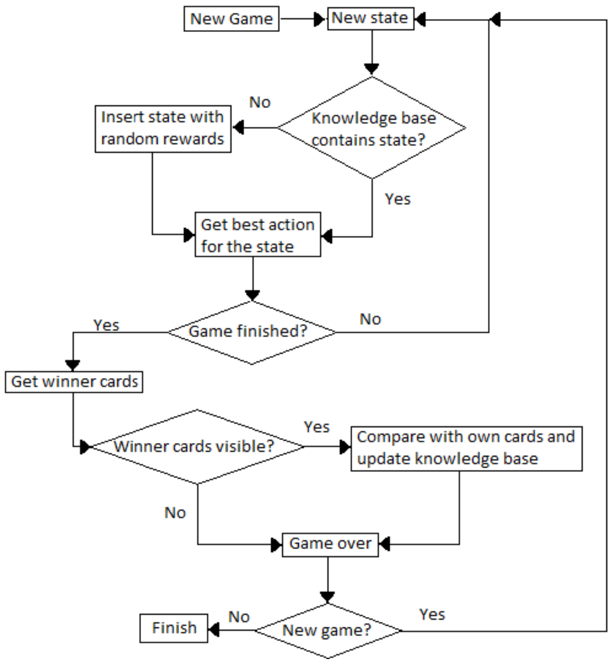
New Game
New state
Knowledge base contains state?
No
Insert state with random rewards
Yes
Get best action for the state
Game finished?
Yes
No
Get winner cards
Winner cards visible?
Yes
Compare with own cards and update knowledge base
No
Game over
New game?
No
Finish
Yes

Fig. 1. Structure of the agent's behavior

4.2 Differences between Agents

Two agents with this structure were implemented: WHSLearner and WHLearner. The only difference between them resides on the reward calculation. Whilst WHSLearner updates the rewards based on the evaluation of the adequacy of the decision, WHLearner considers the actual outcome of the game. Table 2 shows how C and R variables are updated.

Table 2. Decision matrix for WHSLearner agent

Agent		**Agent Action**		
WHSLearner	**WHLearner**	**Fold**	**Call**	**Raise**
Good Choice	Game Won	C↓, R↓	C↑, R↑↑	C↓, R↑↑
Bad Choice	Game Lost	C↑, R↑↑	C↓, R↓↓	C↑, R↓↓↓

5 Tests and Results

To validate this paper's approach several tests were done. In the following sub-sections the experimental procedure and the obtained results are presented.

5.1 Procedure

All tests were performed in a simulated environment. The simulator used was Open Meerkat [17], which is an open source simulator that provides an API that facilitates the implementation and test of Poker agents. Open Meerkat was specially modified to implement the rules described on section 2. The agent was tested against four agents:

- AlwaysCallAgent – an agent that matches every bet;
- AlwaysAllInAgent – an agent that always bets its full bankroll;
- MegaBot [13] – an agent that combines several tactics with a simple heuristic to choose the most appropriate tactic against each kind of opponent;
- SimpleBot [1] – this agent uses opponent modeling and calculates the expected utility of each action based on a Bayesian analysis from simulations.

There were two stages to the agent testing phase. The first stage allowed the agent to build the Q-Table before actually testing its capabilities. Several simulations were run until the weights of actions for each state stabilized. The second stage was meant to test the agent against other agents as described above. There were 100.000 game simulations for each opponent, with seat permutation, to reduce the variance of results, following [1]. The results are displayed in profit evolution charts (figure 2-6).

5.2 Experiments

In the first experiments the opponents were AlwaysCallAgent and AlwaysAllInAgent.

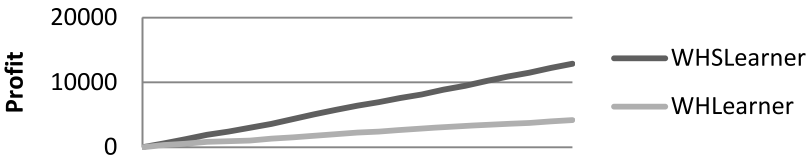

Fig. 2. Profit of both agents when facing the AlwaysCallAgent

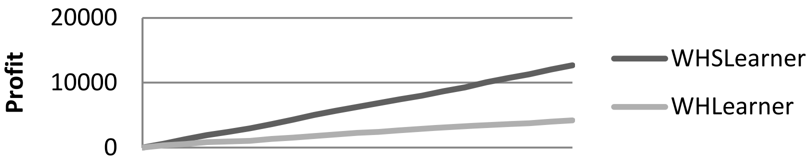

Fig. 3. Profit of both agents when facing the AlwaysAllInAgent

The results presented on Figure 2 and 3 show that this agent's approach is capable of easily beating very basic agents. In both experiments, the WHSLearner performed better. Figures 4 and 5 present the results of experiments where the agent played against more competitive opponents – MegaBot and SimpleBot. As can be seen, against more competitive opponents, the agents still got a positive profit. An interesting fact is that WHLearner performed better than WHSLearner against SimpleBot. This could be due to WHLearner having taken more advantage of the training stage than WHSLearner because of the number of wins of the first was higher than the number of good decisions of the former.

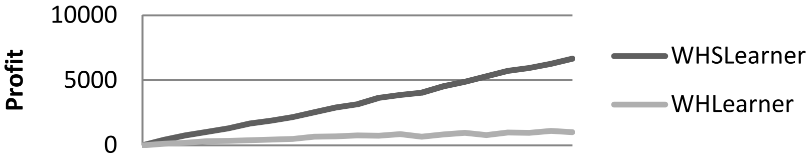

Fig. 4. Profit of both agents when facing the MegaBot

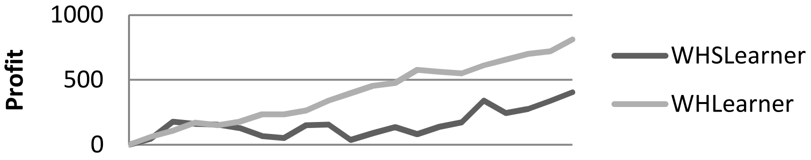

Fig. 5. Profit of both agents when facing the SimpleBot

Finally, the last experiment opposed both agents. Results are shown on figure 6.

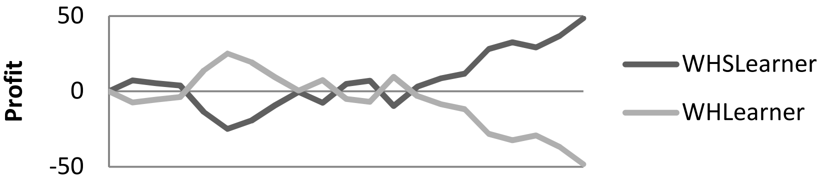

Fig. 6. Profit of both agents when facing each other

Like in most past tests, WHSLearner performed better. This means that adapting rewards using decision assessment is probably better than using the matches' outcome.

6 Conclusions and Future Work

Results showed that this approach is a valid starting point to create a complete Texas Hold'em agent, since the agent outperformed every opponent in all experiments. Another important conclusion can be extracted for the differences between the performance of WHSLearner and WHLearner. In most experiences, WHSLearner performed better, which means that rewarding good decisions maybe better than rewarding good outcomes in reinforcement learning algorithms.

Future work in this project should focus on developing an agent that considers the whole set of rules of Texas Hold'em. Moreover, this approach should be tested with human players for a more proper assessment. Finally more variables can be introduced to better represent the game state.

Acknowledgments. I would like to thank Fundação para a Ciência e a Tecnologia for supporting this work by providing my Ph.D. Scholarship SFRH/BD/71598/2010.

References

1. Billings, D.: Algorithms and Assessment in Computer Poker. Ph.D. University of Alberta, Edmonton, Alberta, Canada (2006)
2. Newborn, M.: Kasparov versus Deep Blue: Computer Chess Comes of Age, 1st edn. Springer (1996)
3. Sklansky, D.: The Theory of Poker: A Professional Poker Player Teaches You How to Think Like One, 4th edn. Two Plus Two (2007)
4. Billings, D.: Computer Poker. M.Sc. University of Alberta, Canada (1995)
5. Davidson, A.: Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment. M.Sc. University Alberta, Edmonton, Alberta, Canada (2002)
6. Schauenberg, T.: Opponent Modeling and Search in Poker. M.Sc. University Alberta, Edmonton, Alberta, Canada (2006)
7. Frank, I., Basin, D., Matsubara, H.: Finding optimal strategies for imperfect information games. In: Proceedings 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, pp. 500–507. American Association for Artificial Intelligence, Menlo Park (1998)
8. Johanson, M.: Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. M.Sc. University Alberta, Edmonton, Alberta, Canada (2007)
9. Gilpin, A., Sandholm, T.: A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In: Proceedings 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, pp. 1453–1454 (2006)
10. Gilpin, A., Sandholm, T.: Better automated abstraction techniques for im-perfect information games, with application to Texas Hold'em poker. In: Proceedings 6th International Joint Conference on Autonomous agents and Multiagent Systems. Article 192, Honolulu, Hawaii, United States, 8 pages (2007)
11. Billings, D., Burch, N., Davidson, A., Holte, R.C., Schaeffer, J., Schauenberg, T., Szafron, D.: Approximating game-theoretic optimal strategies for full-scale poker. In: Proceedings 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, pp. 661–668 (2003)
12. Johanson, M., Bowling, M.: Data Biased Robust Counter Strategies. Journal of Machine Learning Research 5, 264–271 (2009)
13. Teófilo, L.F., Reis, L.P.: Building a No Limit Texas Hold'em Poker Agent Based on Game Logs Using Supervised Learning. In: Kamel, M., Karray, F., Gueaieb, W., Khamis, A. (eds.) AIS 2011. LNCS, vol. 6752, pp. 73–82. Springer, Heidelberg (2011)
14. Kleij, A.A.J.: Monte Carlo Tree Search and Opponent Modeling through Player Clustering in no-limit Texas Hold'em Poker. M.Sc. University of Groningen, Netherlands (2010)
15. Van den Broeck, G., Driessens, K., Ramon, J.: Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS, vol. 5828, pp. 367–381. Springer, Heidelberg (2009)
16. Dahl, F.A.: A Reinforcement Learning Algorithm Applied to Simplified Two-Player Texas Hold'em Poker. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 85–96. Springer, Heidelberg (2001)
17. Open Meerkat Poker Testbed (2012), `http://code.google.com/p/opentestbed/`