

SWE555

RESEARCH PROJECT REPORT

**Review of four artificial
intelligence agents for poker**

Author:

Deniz DIZMAN

Abstract

This report will review four different artificial intelligence agents implemented to play the game of Texas Hold'em poker. Each agent applies a unique approach to the problems presented in creating a powerful poker player. The report will summarize the challenges in these approaches, the implementation and results of the agent if available. The agents being surveyed are GS1 [1], SARTRE [2], AKIREAL Bot [3] and CASPER[4].

1 Introduction

Games are considered a good domain for measuring artificial intelligence due to the simplicity of defining the rules of the game and determining winners but the complexity of a solution or a strategy for winning a player. The domain of poker provides a challenging environment to the subject of artificial intelligence in that it incorporates aspects such as uncertainty, deliberate deception, imperfect information and stochastic events which are not presented in classical games like chess or checkers that AI researcher have been investigating.

The imperfect information of poker comes from the players hidden cards which are not revealed until the end of the game or sometimes never revealed at all. The stochasticity stems from the cards which are dealt from a standard 52 card game deck at certain stages of the game. Each player tries to maximize their winning by defeating the other players. These constraints for the player to make decisions in an uncertain and hostile environment.

Poker has a lot of variants (5 card draw, Omaha, 7 card stud, etc.) but the focus of all the agents investigated in this paper are for the variant called Texas Hold'em poker. Texas Hold'em is a poker variant played with 7 cards and in up to 5 stages. At the beginning stage, called the pre flop stage, each player is dealt 2 hidden cards, also called hole cards or pocket cards. Then the players to the left of the dealer wager in a half and a full bet respectively which are called the small and big blind. The reason of this "blind" betting is to increase the momentum of the game. Then a round of betting takes place in which each player may bet, raise or fold. At any stage a player that folds forfeits from the money gathered in the pot. Up to 3 rounds of consecutive raising may take place. After every player has wagered the same amount of money to the pot, or has folded the game proceeds to the next stage called the flop. During the flop 3 cards are dealt face down on the table that are called the community cards or the board cards. Then another round of betting takes place. After this the game proceeds to the next stage called the turn. During this round one more card is dealt face down and another round of betting takes place. During this round the minimum amount of a bet increases to two full bets (big blinds). After this round the last card is dealt in the river stage and the final round of betting takes place. After this betting round the showdown takes place and all the players still in the game reveal their hole cards and the player with the highest ranking 5 card hand made with the 7 cards wins the pot. In case of a draw the pot is split evenly among the winning players, and the next hand begins from the beginning stage.

The rest of the report will explain the strategies and algorithms used in the mentioned bots.

2 GS1 Agent

2.1 Introduction

The GS1 agent uses a game theoretic approach to heads-up (two player) Texas hold'em. As mentioned, the game of poker is a hostile environment with players trying to maximize their gains. Game theory provides a framework to explain the rational behaviors in such settings. The developers of GS1 have tried to develop computational methods to apply game theory to a real world game of imperfect information. Different from its predecessors that have used game theory such as Spar bot [5], Gs1 does not require very little domain specific knowledge, instead it analyzes the game tree and determines the best abstractions. It also performs on-line and off-line computation, which enables the agent to accurately evaluate strategic situations early in the game when using off line calculation and to perform better abstractions based on a specific part of the game tree when using on-line computation.

2.2 Strategy computation - Pre flop and flop

The GS1 computes these stages off line which involve 2 phases of computation: the automated abstraction and equilibrium approximation.

2.2.1 Automated abstraction

Gs1 uses the gameshrink algorithm [Gilpin and Sandholm, 2005] which design to take as input the description of a game and output an abstraction of the game which can be solved for an equilibrium which can then be used to approximate the equilibrium of the original game. The crudity of the abstraction is controlled by a threshold parameter. In the first round there are $\binom{52}{2} = 1326$ distinct possible hands. However there are only 169 strategically different hands because holding a $A\clubsuit A\spadesuit$ or $A\diamondsuit A\heartsuit$ is in the same equivalence class. Gameshrink automatically discovers this. The next round a positive threshold is used and the strategic nodes are reduced to 2465. Hand evaluation of a 7 card hand is precomputed and stored in a database called *handeval* which has $\binom{52}{7} = 133784560$ entries and is used in many places of the algorithm. Another database *db5* stores the expected number of wins and losses (assuming normal distribution) for five card hands $\binom{52}{2}\binom{50}{3} = 25989600$ corresponding to the hole cards and the flop cards. The *db5* database is used to compare how strategically two hands are similar to each other. These look up databases allow the gameshrink phase to run much faster, that allows a determination for the level of abstraction through trial and error. The best values for abstraction were all 169 distinct hands on the preflop and 2465 classes on the flop.

2.2.2 Equilibrium computation

Only the game that consists of the preflop and flop rounds are considered where the payoffs are computed using an expectation over possible cards for the last hand, but any betting in the final rounds are ignored. GS1 attempts to solve this zero sum game using linear programming, which is a complex task. The difficulty lies in computing the expected payoffs at the leaf nodes of the game tree. Considering the 5 card game history without the bets (bets are not important for computing wins and loses) there are $2.8 * 10^8$ different histories. To obtain each leaf node you need to roll out the remaining cards (990) which makes a total of $2.7 * 10^{13}$ leaf nodes. The GS1 uses a precomputed database called db223 to store the information as explained in the previous section.

Using the abstractions described the GS1 obtains a linear program with 243938 rows, 247107 columns and 101000490 non-zeros. The researchers used the IGOL CPLEX barrier method to solve this LP and obtained near optimal strategies due to the non-lossy abstraction of the preflop hand.

2.3 Strategy computation - Turn and river

The turn is the 4th card revealed on the table. Up to now each player has received a pair of cards and 3 cards are shown on the table. Associated with these rounds are 7 possible betting sequences for the pre flop and 9 possible betting sequences for the flop stage. Additional to these betting histories there are 270725 possible combinations for the community cards. The number of possibilities to consider makes the computation of an optimal strategy hard. GS1 uses a real-time approximation based on the observed history for the current hand for the last two rounds of the hand. This enabled the agent to concentrate on a smaller section of the game tree. Again the agent must perform an automated abstraction and equilibrium computation, but the nature of real-time calculation possesses additional challenges.

2.3.1 Automated abstraction

Again there are some properties that reduce the amount of computation. (1) The appropriate abstraction does not depend on the betting history. (2) Suit isomorphism reduce the combination of the cards to 135408. Although this abstraction step could be performed on line the GS1 implementation chose to do this off line for various reasons, such as allowing the strategy solver more than and being able to choose the best fitting abstraction for a specific combination of board cards from all the available abstractions that can be solved within the time constraint. All 135408 abstractions were computed within a month with a 6 cpu system.

2.3.2 Equilibrium calculation

The probability of the pair of hole cards a player may be holding is calculated using the Bayes theorem taking into account the history and the previous stages

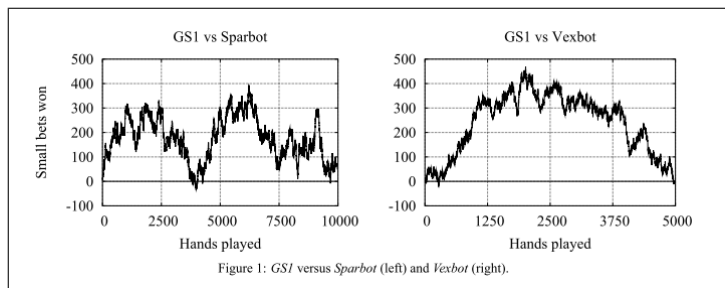
of the game. Letting h denote history, θ denote the possible pair of hole cards, and s_i the strategy of player i , the probability that player i holds the pair θ_i is [1]

$$Pr[\theta_i|h, s_i] = \frac{Pr[h, \theta_i, s_i]Pr[\theta_i]}{Pr[h|s_i]} \quad (1)$$

Once the turn card is dealt, GS1 creates a separate thread to solve the LP. When it is time to act the thread is interrupted and the current solution is given. The thread continues to solve in the background if an optimal solution was not reached to be able to give a better response in case there is a third or fourth betting round. One subtle issue with GS1 occurs when it reaches an information set that later has become an information set with probability zero. For example due to an action call, the agent bets at the time. Later during further analysis of the LP it finds that it should have checked. Now if the opponent re-raises the LP solver cannot offer any guidance because it had bet in the previous round when it should have checked and the agent is in a state that it should have not reached.

2.4 Experimental results

GS1 was tested against Spar bot (Billings et al. 2003) which is also based on game theory. Sparbot computes three betting rounds all in off line mode, and is hardwired never to fold on the preflop stage. In 10,000 hands of poker GS1 won 0.07 small bets per hand on average. The second opponent GS1 was tested on is the Vex bot agent by Billings et al. 2004. Vex bot uses game tree search with opponent modeling and is able to adapt to a fixed strategy like in GS1 and can improve it's strategy. After 5000 hands of simulation the match ended up in a tie as would be expected by a game theoretic approach. Figure [1] depicts the winnings in the games.



3 SARTRE Agent

3.1 Introduction

SARTRE [2] is a case based reasoning system that uses a memory based approach to heads up (2 player) limit Texas Hold'em poker. The agent uses hands played by previous players and uses them to make decisions. Instead of using a system to solve the game theoretic equation this agent tries to re-use previous hands played by strong players to achieve a similar performance. The knowledge base of SARTRE is constructed from the hand histories of previous games from the AAAI CPC (computer poker competition). In 2008 the University of Alberta's Hyperborean-eq won the championship which is a fixed near equilibrium player. SARTRE knowledge base was constructed from the games Hyperborean-eq had played.

3.2 System overview

SARTRE searches for similar cases in its knowledge base that would fit the current situation. There are three factors that were hand picked by its authors that it uses:

1. The previous betting for the current hand
2. The current strength of SARTRE hand
3. The texture of the board

3.2.1 The previous betting for the current hand

Each betting round is represented as a path in a betting tree, which enumerates all the betting combinations up to a certain point in the hand. A path within this tree represents the choices made. Given two different trees the authors tried to compute the similarity between these two paths. A similarity value between 1.0 and 0.0 is assigned where 1.0 is an exact match. The figure below depicts a betting tree where c represents a bet call, f is a fold and r is a raise [2].

3.2.2 The current strength of SARTRE hand

The hands of the agent is mapped into an class of available poker hands which as no-pair, one pair, two pair, three of a kind, straight, flush, full house, four of a kind, and a straight flush. During the turn and river stages of the game the players hand has a chance to improve since not all the cards have been dealt out, these states are called drawing hands. SARTRE considers two types of drawing hands: Straight draws and flush draws. The hand categories that SARTRE uses were predetermined by the authors. Some more examples of the categories that SARTRE uses to distinguish hand strength is over-cards which indicate that the hole cards of the agent are higher than any card on the board and no pair have been made, and ace-high-flush-draw-uses-both which indicated that SARTRE

a nash equilibrium. The matches against FellOmen2 were conducted using the AAAI CPC poker server version 2.3.1 with 6 separate duplicate matches each 6000 hands each making a total of 36,000 hands. Duplicate game are played when N hands are played in forward direction then, the agents memories are reset and the game is played in reverse order, i.e the agents play with the cards of the other agent. This is done to reduce the variance. The matches between Bluffbot and SARTRE were conducted on the commercial application poker academy (<http://www.poker-academy.com>) which does not support duplicate games. A total of 30,000 hands were played between them.

The results against FellOmen2 were -2.92 ± 0.5 big blinds per 100 hands, which translates to -11.60 ± 2 for a game of 2/4 poker. The results against BluffBot were $+7.48$ BB per 100 hands.

The authors of SARTRE conclude that their agent has yet not reached the level of player of it's role model hyperborean-eq as it was not profitable against FellOmen2 but hyperborean-eq was. They state following reasons for this:

1. The hands strength feature is not sophisticated enough and maps dissimilar hands into the same category which results in information being lost.
2. The case selection is coarse in many cases. For a random match 10 percent were unmatchable and a default action of calling was selected.

4 AKI-RealBot Agent

4.1 Introduction

AKIReal bot [3] is an exploitative ring game (multi player) limit Texas hold'em poker agent that uses Monte Carlo tree search to evaluate and make decisions. It tries to find weaknesses in opponent plays and unlike nash-equilibrium bots we have talked about, it's aim is not at worst to break even but to exploit the opponent to maximize winnings.

4.2 Decision Engine

4.2.1 Monte Carlo Search

The Monte Carlo methods [Metropolis and Ulam, 1949] are a commonly used as approaches in scientific areas. In game playing context it means that instead of searching the whole game tree, random paths are chosen in the tree. When compared to an evaluation function which also tries to limit the search space, Monte Carlo methods limit the search breadth at each node, and use a probabilistic approach at decision nodes. In the game of poker there three possible actions at each decision node: call, bet and fold. AkiRealBot typically runs a simulation and calculates the expected values (EV) for each action. These EV's are calculated by applying independent searches for the call and for the raise action. The simulation is limited by a timer module which cuts the simulation

when the time runs out. Since more simulation rounds mean a better EV a multi threaded approach was taken by the authors. The Monte Carlo search for AkiReal Bot is not based on a normal distribution but is influenced by the actions that the players have taken during the hand. For this purpose it collects information about the players fold, call and bet actions and builds an opponent model.

4.2.2 Post Decision Processing

AkiReal Bot uses a post processor on the Monte Carlo engines decision to be adaptive to different kinds of players and to exploit any weaknesses they have. The exploitation is considered in two different factors:

As long as the EV of folding is lower than the EV of calling or raising it makes sense to stay in the game. A more aggressive strategy would be to stay in the game even if the EV of folding minus a factor δ is lower for a positive value if δ . AkiReal Bot maintains a statistic over 500 hands against the opponent to calculate a lower bound. If the agent W has lost 0.25 SB (small bets) to AkiReal Bot over the 500 hands then the factor d is calculated as $0.5 \times 500 = 250$. If d is in the range $[-100;100]$ then aggressive playing style is assumed and the factor δ is calculated as

$$\delta(d) = \max(-0.6, -0.2 \times (1.2)^d) \tag{2}$$

To calculate the upper bound which will force the agent to raise even if $EV(\text{call}) > EV(\text{raise})$ is calculated as

$$\rho(d) = \min(1.5, 1.5 \times (0.95)^d) \tag{3}$$

The upper bound $\rho(0) = 1.5$ for $d = 0$ which is 1.5 SB is a very confident EV and is taken as the upper limit for the upper bound. The aggressive raise value is not influenced by loses against agent W but will converge to 0 for wins which results in a very aggressive player.

4.3 Opponent Modeling

AkiReal bot employs an opponent model which treats every player as a straight forward and rational player to begin with, which means that is assumes that players will raise with a strong hand, call with a mediocre hand and fold with a weak hand. It has two different functions to assign cards to an opponents hole cards which are used at different stages of the game.

In the pre-flop stage, it is assumed that the actions of the player are based on their hole cards. AkiReal Bot divides the beginning hole cards into 5 buckets of strength. The first bucket has the weakest card class and the last have the strongest. The authors have assign the following probability distributions to the buckets $p(U_0) = 0.65, p(U_1) = 0.14, p(U_2) = 0.11, p(U_3) = 0.07, p(U_4) = 0.03$ If for example an opponent W would call a raise, then the upper bound is set to the strongest bucket, and the lower bound is calculated as $l = c + f$. If we

assume $f = 0.72$ and $c = 0.2$, then W raises only 9% of the cases. This implies that it would raise with the top 9% of its hole cards in which case the lower bound would be set to the fourth bucket. After the boundaries are set the hand for the player is selected at random from the buckets.

In the post flop stage the basic difference is that the opponent chooses his actions based on the board cards. According to the data gather from the opponent two different methods are used for card assignment

1. assignTopPair - increase the strength of the hole cards by assigning a card that would make the player have the top pair.
2. assignNutCard - increase the strength so that the player would have the highest possible hand.

The second hole card is assigned at random. This method of card assignment has the draw back that it may underestimate the opponent cards, for example if there are 3 suited cards on the board, assignNutCard may not assign 2 more of the same suit to the player.

4.4 Experimental Results

AkiReal Bot entered the CPC in 2008 and finished at second place in the 6 player limit ring game tournament. The competing entries were Hyperborean08-ring a.k.a Poki0 (University of Alberta), DCU (Dublin City University), CMUR-ing(Carnigie Mellon University), GUS6(Georgia State University), MCBotUltra, AkiRealBot and 2 indepdent entries from T.U. Darmstadt. Among all 6 players 84 matches were played with different seating permutations for a total of 504000 hands, and the winners were determined by the accumulated results of winnings over all the games. A significant observation is that AkiRealBot only manages to defeat three opponents and loses to two. But it defeats GUS6 so badly that overall it places as second. This shows that AkiReal Bot can really exploit weak players but cannot compete with stronger and solid players.

5 CASPER Agent

5.1 Introduction

CASPER is a cased based reasoning agent like SARTRE that uses a previous history of hands to make poker decision. The improvement over previous CBR systems is that CASPER incorporates more elements such as the state of the table, betting positions, etc. to make the decision.

5.2 System overview

When it is CASPER's turn to act the agent evaluates the current state of the game and constructs a target representation. The representation includes factors of the game such as CASPER hand strength, how many opponents are

in the pot, how many opponents are to act, and how much money is in the pot. After this case is constructed CASPER consults its knowledge base and tries to find similar scenarios. CASPER uses the k-nearest neighbor algorithm to match target cases against its case. The knowledge base of CASPER was constructed from games player the different bots provided with the commercial software poker academy. Each decision during the 7000 hands were recorded into CASPER’s knowledge base.

5.3 Case representation

CASPER searches a separate knowledge base for each stage the hand, pre-flop, flop, turn, river. The cases indexed are believed to the prediction of the further game progression and the outcome of the current stage is found by local similarity for each feature. Each case has a single outcome which is the betting action. The hand strength feature is calculated differently for pre-flop and post-flop games. In the pre-flop stage each of the possible 169 card combinations are numbered from 1 to 169 with 1 being the strongest pair which is a pair of aces and 169 being 7 and 2 offsuit. The strength after the flop is calculated by enumerating all possible hole cards for an opponent and computing how many of these hands is stronger than, equal to or worse than CASPER’s hand.

5.4 Case retrieval

Once the target case has been constructed CASPER scans the knowledge base to find a similar case. Each feature has a local similarity metric associated with it, where 1.0 denotes an exact match and 0.0 entirely dissimilar. CASPER uses two types of similarity metrics. The first one is the standard Euclidean distance function given by [4]

$$s_i = 1 - \left(\frac{|x_1 - x_2|}{MAX_DIFF} \right) \quad (4)$$

where x_1 is the target and x_2 is the case value and MAXDIFF is the greatest difference in the values.

For some features such as the bets to call the above metric produces major changes in output for a small change in the input. For this reason an exponential decay function has been used in some features [4]

$$s_i = e^{-k(|x_1 - x_2|)} \quad (5)$$

where x_1 is the target value and x_2 is the case value and k is a coefficient that controls the rate of decay.

Global similarity is computed as a weighted average of the local similarities with the following formula [4]

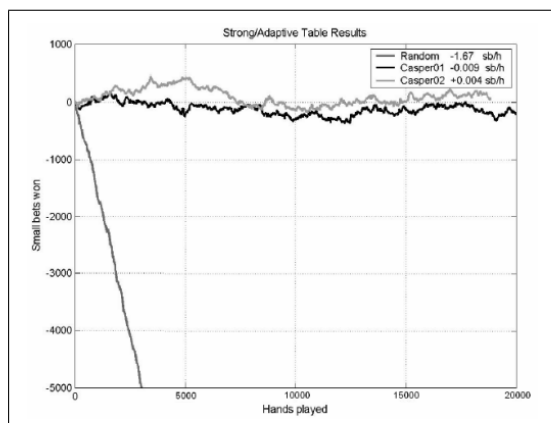
$$\sum_{i=1}^n \frac{w_i x_i}{w_i} \quad (6)$$

where x_i is the local similarity metric in $[0;1.0]$ and w_i is the weight assigned to that metric in the range $[0;100]$

After the calculation of the of the similarity values for each case, they order sorted in descending order using quick sort and all cases exceed a threshold of 97% similarity are considered as matches. Each action is summed up and divided by the total number of similar cases to form the probability triple $\text{pr}(f,c,r)$ which gives the probability of folding, calling or raising. If no cases exceed the threshold can be found than the top 20 cases are chosen.

5.5 Experimental results

CASPER was tested against the poker acedemy bots, from which it actually constructed its knowledge base and against other bots. Against adaptive bots which use opponent modeling CASPER01 had a loss of 0.09\$ per hand whereas CASPER02, a slight improvement over CASPER01 with a larger knowledge base had a win of 0.04\$ per hand. A poker bot that makes random decision were included as a base line for the testing. The figure below shows the match results [4]



6 Conclusion

Four agents of limit Texas hold'em poker were examined and 3 different approaches to the problem were considered. As of today poker still remains an unsolved game, but with artificial intelligence agents challenging the worlds top players, the field is promising area in research.

References

- [1] Andrew Gilpin and Tuomas Sandholm, *A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation*

Computer Science Department Carnegie Mellon University, 2006.

- [2] Jonathan Rubin and Ian Watson *A Memory-Based Approach to Two-Player Texas Hold'em* Department of Computer Science University of Auckland, New Zealand
- [3] Immanuel Schweizer, Kamill Panitzek, Sang-Hyeun Park and Johannes Furnkranz *An Exploitative Monte-Carlo Poker Agent* TU Darmstadt - Knowledge Engineering Group
- [4] Ian Watson, Song Lee, Jonathan Rubin Stefan Wender *Improving a Case-Based Texas Holdem Poker Bot* Dept of Computer Science, University of Auckland, Auckland, New Zealand
- [5] Billings, D.; Burch, N.; Davidson, A.; Holte, R.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. *Approximating game-theoretic optimal strategies for full-scale poker. In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*. 2003