SAPIENZA
UNIVERSITÀ DI ROMA

Dipartimento di Informatica e Sistemistica
Antonio Ruberti

Master in Artificial Intelligence and Robotics

Report for the project in Artificial Intelligence and Games

Nash Equilibrium and Game Theory on Poker Texas Holdem

STUDENT:
Giovanni Murru

PROFESSOR:
Prof. Marco Schaerf

SUMMER 2011

Contents

1 Introduction

The use of games as research domain plays a significant role in the field of artificial intelligence. Game competitions are organized by the researchers, resulting usually in improvements at the state-of-the-art.

Due to its properties of imperfect information and stochasticity the game of Poker has been identified as a valuable domain for current artificial intelligence academic research. This has led to the investigation of a wide new range of approaches and algorithms. Conferences such as AAAI and IJCAI since 2006 established annual computer poker competitions and resulted in the development of a large number of computerized poker agents.

In *The Games Computers (and People) Play*, Schaeffer identifies poker as a game where human supremacy may soon be challenged. As a matter of fact poker is one of the most complicated problem to solve.

Poker is a class of multiplayer card games with many variants which however present several similarities: they use a standard playing card deck, there are betting rounds and the standard five-card hand ranking system is used to determine the winner. Between the numerous variants of poker, Texas Hold'em is without any doubt the most popular one. This variant include two typologies of game: *limit* and *no-limit*. The differences between these two variants will be explained later.

During 2003 the number of Texas Hold'em players increased exponentially. In the same year Chris Moneymaker qualified for WSOP (World Series of Poker), the biggest live event in the world of Texas Hold'em, through a $39 online satellite tournament at PokerStars. The cost for the other participants was $10.000. Chris won the $2.5 Million event's first price. After the memorable victory Moneymaker left his job of accountant and started to travel around the world to play in more large buy-in tournaments. In 2005 Chris Moneymaker wrote a book titled *Moneymaker: How an Amateur Poker Player Turned $40 into $2.5 Million at the World Series of Poker*. After this event online tournaments became so popular and began to spread all over the world, that even Italy legalized online poker tournaments in September of 2008.

Because of its popularity and its features Texas Hold'em poker has become the main testbed for evaluating algorithms in extensive-form games. It also contains an enormous strategy space, imperfect information, stochastic events, all the elements that characterize most

of the challenging problems in multi-agent systems and game theory. Several poker agents are presented in the following report, using concepts such as Nash equilibrium and best response.

2 The Rules of Texas Hold'em

The Game of Texas Hold'em is played using a deck of 52 French cards involving a number of two up to ten players. Every player receives 2 hole cards, which only they can see. The game is divided in four stages, every of which include a betting round.

2.1 Betting Actions

The possible betting actions of poker are described as follows:

Fold: This action implies the player wants to leave the game.

Check/Call: Check action is when a player express is willing to remain in the game but is not forced to commit further chips because no raise or bet has been done in advance. A player does a Call if required to put an amount of chips in the pot to cover the current bet.

Bet/Raise: The bet is done when a player wants to invest further chips on the pot. The Raise is a bet done in reply to a previous bet to manifest an intention to invest further chips on the pot.

Jam Also know as **all-in** consists in putting all remaining chips on the pot, even if that amount is smaller than the amount needed to call.

2.2 Game Steps

As already said the game evolution can be separated in 4 main stages: Preflop, Flop, Turn and River.

Preflop: One player is selected to be the button (or dealer), and the next two players following a clockwise order (to the left of the dealer) are respectively named **small blind** (SB) and **big blind** (BB). In the case only two players are in the game (heads-up variant), the dealer plays the SB role too. Two forced bets are contributed to the pot before dealing the cards by SB and BB.

The big blind is typically double amount of chips that of the small blind. Once BB and SB put the chips into the pot 2 hole cards are given to the players by the dealer. Then a betting round begins starting from the player to the left of the BB, known as **under the gun**.

Flop: The flop round begins with three cards dealt face up (one card is discarded) from the deck by the button in to the game board. Then a second betting round begins. The three cards are named flop and they can be used by all the players to make the final point.

Turn The turn refers to the third phase of the game, when another card is dealt face up by the dealer. After that a betting round takes place.

River The last phase of the game consists in dealing face up a fifth community card followed by a final round of betting. After that the players still in-game (the ones that have not folded) are requested to face up their cards: an act known as **showdown**. The player with the best five-card hand, constructed from his two hole cards and the five community cards, wins the pot.

2.3 Card Ranking

The rank of the five-card hands is based on the probability of the hand to appear during the game.

1. **Straight flush:**	*e.g.*	6♣	7♣	8♣	9♣	T♣
2. **Four of a kind:**	*e.g.*	K♠	K♠	K♠	K♠	T♡
3. **Full House**	*e.g.*	3♠	3♠	3♠	Q♡	Q♡
4. **Flush**	*e.g.*	A♢	3♢	8♢	J♢	K♢
5. **Straight**	*e.g.*	2♣	3♢	4♣	5♡	6♠
6. **Three of a kind**	*e.g.*	J♢	J♢	J♢	K♠	T♠
7. **Two pair**	*e.g.*	7♡	7♡	6♢	K♣	K♣
8. **One pair**	*e.g.*	7♣	7♣	3♠	K♢	J♢
9. **High card**	*e.g.*	A♡	7♢	3♠	K♢	J♢

2.4 Limit, No Limit and Tournaments

Texas Hold'em presents two main modalities of play: limit and no-limit. The only difference between them is that in no-limit it is pos-

sible to do a jam (to go all-in). On the contrary the limit variant restrict player's bet to a fixed maximum amount of chips.

No-limit variant is without any doubt the most popular and interesting. In fact it is the one played in international competitions such as WSOP, the one won by Chris Moneymaker in 2003.

Even if the rules of the game are essential and simple, the same can't be said about the strategy influencing hands' evolution. Texas Hold'em can be played using chips with a correspondent value in real money, or can use the very common tournament style, in which each player pays a fixed amount of money to participate in the tournament. In this later case the chips has no real value in money, but the player is eliminated from the tournament if he runs out of chips. Three prices for the first, the second and the third ranked players are usually given as award.

Tournament are an extremely popular form of poker. Usually the blinds increase every five or ten minutes in online tournaments. At some point of the game, when the blinds are high enough it can be introduced a fixed mandatory bet for each player in the game, called **ante**.

3 Decision Factors

What is more convenient to do in order to win? That's the question decision factors try to solve. When the player of Texas Hold'em has to take a decision he typically use some metrics acquired with experience and mathematics. The main elements useful to take a decision in Texas Hold'em are:

1. Hole cards
2. Player's position
3. Stacks
4. Number of players
5. Pot odds
6. Opponent's strategy

Hole cards: Hole cards are the two hidden cards that each player received during the first stage of the game. To evaluate the value

of the starting hand one of the most famous writers of books about poker, David Sklansky, created a table which divides the best hands in groups:

Group 1:	AA	KK	QQ	JJ	AKs		
Group 2:	TT	AQs	AJs	KQs	AK		
Group 3:	99	JTs	QJs	KJs	ATs	AQ	
Group 4:	T9s KTs	KQ	88	QTs	98s	J9s	AJ
Group 5:	77 76s	87s 97s	Q9s Axs	T8s 65s	KJ	QJ	JT
Group 6:	66 K9s	AT J8s	55 75s	86s	KT	QT	54s
Group 7:	44 43s	J9 22	64s Kxs	T9 T7s	53s Q8s	33	98
Group 8:	87 85s T8	A9 J8 43	Q9 J7s	76 65	42s 54	32s 74s	96s K9

Where **s** stands for suited (cards of the same suit) and **x** stands for a generic low card. With the Sklansky interpratation a player can help himself to take a decision about folding or not. Let's say a player should never fold if his hand is in the first group. As regard the other groups other decision factors should be considered.

Player's position: Another important decision factor is the position of the player. Speaking as the last one makes a great advantage for the player. In this position the player can safely raise with hands of the group 2, while for the player under the gun (the first to speak) it should be better a limp (entering in the pot only covering the amount of the big blind). Starting from the group 3 we should be careful, specially if the number of the player is great (e.g 10 players).

Stack: They are the amount of chips of each player. This is a very

important decision factor because it influences in various ways the style of game of the player and his inclination in betting or folding. In a tournament players start with the same amount of chips. However soon the stacks become to differ and the choice of the players are strongly influenced by the amount of the chips in the stacks.

Number of players: If the number of players decrease even a hand of the 4th group can be raised. Hence the number of players is an important decision factor too. In fact as the number of players decrease the probability that one player receives a strong hand decreases too.

Pot odds: They are the ratio of the current size of the pot to the cost of the contemplated call. For example if the pot contains \$100 and a player must call \$10 to stay in the hand, then the player has 10:1 pot odds. Pot odds are often compared to the probability of winning a hand with a future card in order to estimate the call's expected value. For this reason pot odds are usually converted to and expressed as probabilities. In the previous example the total pot after the call is \$110 and the percentage of the bet is approximately about 9% of the total pot. It's all about percentage and probabilities of completing an hand.

Opponents strategy The peculiarity of a human player is its intrinsic diversity. However we can generalize and identify 4 big categories of players:

1. **Loose-passive:** The less dangerous players. They play too much hands and they never raise or bet. They try to realize impossible hands and they are minded to call in order to arrive at the showdown.
2. **Loose-aggressive:** They are more dangerous than loose-passive but not too much. They tend to enter in many pots but they are willing to raise even with low hands. They are unpredictable and hence a little dangerous.
3. **Tight-passive:** They play few hands and when they do it they call (never raise).
4. **Tight-aggressive:** They play few hands as like as tight-passive, but when they have the good hand they will raise and maybe raise again. They are the most dangerous.

As easy to imagine, good players are those ones that can adapt their style of play to one of this category depending on the current situation.

4 Extensive Form Games

An extensive-form game is a general model of multiagent sequential decision-making with imperfect information. In other words it is a game that models interactions between multiple autonomous agents by describing sequential decision-making scenarios, in situations where imperfect information and non-determinism may emerge.

Extensive-form games consists mainly of a game tree, where each internal node (choice) has an associated player that takes decision at that node, and each terminal node has associated utilities for the players. Game states are partitioned in information sets composed by indistinguishable states for the player. The directed edges leaving a choice node represent the possible actions for that player. Special nodes (chance) are used to denote stochastic events. In this case the directed edges leaving a chance node delineate the possible chance outcomes.

Extensive games can model a wide collection of strategic decision-making scenarios and because of that they have been used to study poker. Formally an **extensive-form game** consists of:

- N players
- A set of H histories, composed by a sequence of valid actions for the game.
- A set $T \subseteq H$ of terminal histories, that represent the end of the game in the sense that no more actions are allowed.
- A player function P to determine the turn of the player at each point in the game.
- A function f_c defining the chance probabilities.
- An information partition $\mathbf{I}_i$ for each player i consisting of all of the information sets I_i for that player.
- A utility function u_i for each player i that assigns a real value to each terminal history, corresponding to the payoff for that player when the terminal history is reached.

4.1 Computer Poker Strategies, Performance Metrics and Evaluation

In this section we will briefly describe the several types of performance measurements and agent evaluations that are generally used in computer poker. Let's first consider the type of strategies a poker agent could use.

4.2 Strategies

Strategies are classified in pure and mixed. A pure strategy is one which is characterized by choosing a unique action during the game. For example in a simplified poker there are three possible pure strategies: always fold, always bet (or raise), and always check (or call).

However mixed strategies, which assign a probability value to each pure strategy, are usually preferred. In the following we simply refer to them as strategies.

Imperfect information games have some game states that are not differentiable due to hidden information by the agents. For a given agent, the set including those indistinguishable game states is called *information set*.

To play an extensive form game each player must have a strategy, which in this context refers to a mapping between information sets (set of indistinguishable game states) and the actions that an agent will take at those information sets.

Moreover an agent's strategy can be defined as a probability triple over the possible actions at every information set: the probability to fold, check/call or bet/raise (f, c, r). Note that $f + c + r = 1$. Strategies can be static or adaptive over time.

We can generalize a strategy profile σ as a set of strategies $\sigma_i(I, a)$, one for each player i, expressed as probabilities that a player will take action a when its game state is in information set I. Each strategy profile has a correspondent utility $u_i(\sigma)$.

An equilibrium strategy, also known as Nash Equilibrium, is generally considered a solution to the game because finding a Nash Equilibrium is equivalent to determining the optimal combination of probability values that make up a mixed strategy [4].

Game theory is a branch of mathematics that deals with decision-making in situations in which two or more players have competing interests.

Nash equilibrium is a game theory's concept involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing his own strategy unilaterally, because this fact would result in a negative impact on the expected value for them. If each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium.

Another concept coming from game theory is the **best response**. It can be defined as a strategy that obtains the highest player's expected utility against the set of all other strategy profile.

In terms of utility, Nash equilibrium is a strategy such that no player can improve its utility by changing it.

Strategies in Nash equilibrium are a best response for the other strategies in the profile.

A **zero-sum game** is one in which the sum of the utilities associated to the players is null. For a zero-sum two-player game every strategy for a given player in a Nash equilibrium strategy profile has the same utility and is a best response to all the strategies in equilibrium profiles for the other player. As a matter of reason if a player plays a strategy from one equilibrium strategy profile, the other player cannot gain utility by playing a strategy from a different equilibrium strategy profile. This result however is not true for multiplayer games or in case of non-zero-sum.

Solutions to the Nash equilibrium can be found as minimization problems with technique such as linear programming. Unfortunately this one works only with small games. Texas hold'em is a large extensive game and as consequence it is intractable to compute a real Nash equilibrium. However by applying some abstractions on the game it is possible to compute a ϵ-Nash equilibrium, a strategy profile in which no player can increase its utility by more than ϵ by unilaterally changing its strategy.

5 Abstractions

The most interesting real-world games have a so large tree-space that even the best algorithms have no hope of computing an equilibrium solution. Limit Texas Hold'em, for example, has about 10^{18} states. The no limit version has even more states: 10^{71}. In order to deal with

these huge spaces, abstractions are introduced.

The main idea beside abstractions is to collapse several information sets of the original game into single information sets of the abstracted game, often referred as buckets. There exists many types of abstractions. One of the most popular is **card abstraction**, which is performed through a bucketing of the hand strength. Two common variants are *percentile bucketing*, which places more or less an equal amount of hands into each bucket, and *uniform bucketing*, which performs a homogeneous partitioning of the hands' strength in the buckets.

Metrics typically used should be coherent with hole card's power. As example we can take hand strength expected value E[HS], or squared hand strength's expected value E[HS^2].

Adopting this simplification hands belonging to the same bucket are played in a identical way, resulting in considerable reduction of the game tree.

Expectation-based abstraction are derived by bucketing hands considering the expected hand strength. Nested bucketing performs a first bucketing based on E[HS^2], and after that a second bucketing splits each of the first buckets based on standard E[HS]. The already explained *percentile bucketing* is an automated technique that uniformly distributes the hands inside a certain number of buckets. Hence using this type of bucketing each slot approximately contains the same amount of hands. *History bucketing* takes advantage of the fact that some buckets have greater probability of transitioning to particular buckets than others. As a matter of fact it is more likely that if a hand belongs to a high hand strength in a certain round, then in the next one it will transition to a high hand strength bucket. Analogously if the hand belongs to a low hand strength bucket, it will remain in a low hand strength bucket with high probability.

Unlikely from expectation-based abstractions, potential-aware automated abstractions make use of multi-dimensional histograms to assign similar hands to the same bucket.

If the abstraction is constructed so that all players are able to remember all their previous actions (never forget information once it is acquired), the abstraction is named *perfect recall*. To further reduce game space *imperfect recall* abstraction may be introduced. In fact the old information relative to previous observations is forgot and state space is saved. Imperfect recall permits to assign more computational resource to the current betting round.

Other abstractions involving the betting round are the *betting round reduction*, which consists in reducing the maximum number of bet (raise) in each betting round, or the more radical *elimination of betting rounds*, which is an abstraction that involves the elimination of the entire betting rounds (e.g. a game of poker ending at the turn).

As a matter of fact these lossy abstractions bring only to approximate equilibrium solutions. However the solutions computed using abstracted games can be mapped to strategy profiles in the original games.

Another type of abstraction used to reduce the game tree is the betting abstraction, in which some betting options are eliminated.

5.1 Automated Action Abstraction

John Hawkin, Robert Holte and Duane Szafron of University of Alaberta developed in [1] a technique that abstracts a game's action space. It is important to understand that a successful abstraction technique is one that creates a smaller game retaining the important strategic features of the original game.

The problem of abstraction in the action space can be split into two parts:

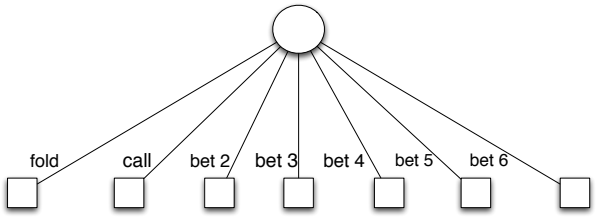
- **Betting abstraction**, choosing which bet size to remove from the game.
- **Translation problem**, determining how to react to opponent bet sizes.

An example of simple action abstraction is an agent that can bet only a fraction of the pot size or all its chips.

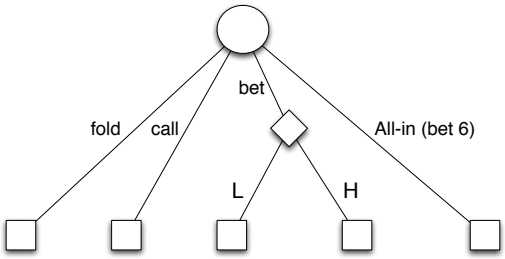
Researchers designed a multi-player betting game transformation that can be applied to domains with actions (bets) that have associated real or multi-valued parameters (the amount of the bet in size).

As we can see in Figure 5.1, most betting actions are removed at each decision point, and some of them are joined in a individual new action that is called bet. An action All-In (betting every chip in the player's stack) is introduced. The bet action is differentiated into two possible choices: low bet (L) and high bet (H). The low and high bet can depend on the player (on its stack), but it's crystal clear that L should be always less than H, and be at least a minimum bet.

The decision of whether to bet L or H is private to the player that chose it. In Figure 5.1 player one's decision point is represented by



(a) Regular no-limit game



(b) Multiplayer betting game

Figure 5.1: Decision point in a no-limit game, and the multiplayer betting version of the same decision point.

the circle, while the square represent the decision point of player two. The diamond denotes the bet sizing player.

Denoting $P(H)$ and $P(L)$ respectively as the probability of betting H and L, we can assert that $P(H) + P(L) = 1$. Then, the effective bet size function is defined as:

$$B(P(H)) = (1 - P(H))L + P(H)H \tag{5.1}$$

Lemma 1 *For any fixed values of L and H and probability $P(H)_i$ of player i betting H, the expected value of the pot size after that bet for all players j such that $j \neq i$ is equal to $p + B(P(H)_i)p$, where p is the pot size before the bet.*

Theorem 1 *For any given strategy profile σ of a multiplayer betting game, $u_i(\sigma)$ where $i = 1$ or $i = 2$ is equivalent to $u_i(\sigma')$ in the abstract no-limit game with the same stack sizes where at each corresponding decision point only bet sizes of $s = B(PH)$ and All-in are allowed, and $\sigma'_i = \sigma_i$, for $i = 1, 2$.*

Theorem 1 allows a mapping between a given strategy profile σ for a given multiplayer betting game and a strategy profile σ' for the abstract poker game. The choice of the abstraction is part of

player's strategy; in fact each different choice of $P(H_i)$ correspond to a different abstraction.

5.1.1 Nash equilibrium solution of half-street no-limit Kuhn Poker

In half-street Kuhn poker there are 2 non-dominated strategy parameters:

- $P(b|j)$, the probability that a player bets having a Jack.
- $P(c|q)$, the probability that a player calls having a Queen.

In case of limit version, fixed bet size s and antes of 0.5, a strategy profile is a Nash equilibrium if and only if:

$$P(b|j) = \frac{1 - P(c|q)}{2} = \frac{s}{s+1} \tag{5.2}$$

However this sole condition is not enough for the multiplayer betting version. In such case all bets can be denoted as the effective bet size $s = B(P(H))$. Furthermore $P(H)$ should be such that the bet sizing player cannot gain by choosing a different value for the bet, which means that:

$$u_1(B(x)) - u_1(s) \leq 0 \qquad \text{for all } x \in [0, 1] \tag{5.3}$$

For this to be true we must have that:

$$P(b|j) = \frac{P(c|q)}{1 + P(c|q)} \tag{5.4}$$

A Nash equilibrium is found if and only if both equations 5.2 and 5.4 are satisfied. That brings to the solution where $P(c|q) = \sqrt{2} - 1$. Replacing this value in equation 5.2, the solution $s = \sqrt{2} - 1$ is found.

Hence a Nash equilibrium strategy profile for the multiplayer half-street Kuhn poker game is found if we define $P(H)$ such that $B(P(H)) = s = \sqrt{2} - 1$.

Researchers in [1] adapted the algorithm to no-limit version of this game, finding out the value $B(P(H)) = 0.414$ was the correct bet size to achieve a Nash equilibrium strategy profile.

5.1.2 Finding bet sizes in larger games using abstractions

However results in previous section are computed only as baseline since analytical solutions are well-known. In the case of larger games such as Leduc poker, abstractions are needed in order to find a approximated equilibrium solution using existing techniques as CFR algorithm. A common betting abstraction for no-limit games is to allow call, fold, bet and all-in actions everywhere.

6 Exploitative Counter Strategies

Nash equilibrium solutions are robust and static strategies that limit their own exploitability by maximizing a minimum outcome against a perfect opponent. Despite that, in the case of weak opponents, these kind of strategy may perform bad. An exploitive strategy is one that tries to benefit from opponent's weakness. Introducing exploit strategy deviates from equilibrium point and often lead to higher payoffs. However opponents may play a certain strategy for several iterations to trick the exploiter, which will cause the exploiter to become the exploited.

To be safe we must use algorithms that are exploitable only to some extent, otherwise a lower payoff than the one gained using the equilibrium strategy may result.

In order to exploit the opponent, we need to construct a model of his behavior. This is typically performed offline. We can for example start playing the equilibrium for several times so that we can obtain a valuable number of samples of the opponent's play.

Exploitation of a set of different opponents can be performed using counter-strategies' teams, whose members must be able to exploit all the different styles of opponents.

6.1 An efficient real time opponent modeling algorithm

Deviation-Based Best Response (DBBR) is an algorithm developed and presented by the researchers Sam Ganzfried and Tuomas Sandholm in their recent paper [2] about game theory-based opponent modeling.

DBBR builds the model based on the deviations of the opponent's action from the equilibrium strategy. The algorithm works in first analysis trying to find an approximate equilibrium σ^* offline.

Algorithm 1 High-level overview of *DBBR*

Compute an approximate equilibrium of the game.
Mantain counters from observing opponent's play throughout the match.
for $n = 1$ **to** $|PH_{-i}|$ **do**
    Compute posterior action probabilities at n.
    Compute posterior bucket probabilities at n.
    Compute full model of opponent's strategy at n.
end for
return Best response to the opponent model.

Then, a recorded public history of opponent's action frequencies is used to compute the opponent posterior action probabilities $\alpha_{n,a}$. These values are computed observing how often the opponent chooses action a at each public history set $n \in PH_{-i}$. The elements of PH_{-i} are ordered according to breadth-first-search (BFS).

$$\alpha_{n,a} = \frac{p^*_{n,a} \cdot N_{prior} + c_{n,a}}{N_{prior} + \sum_{a'} c_{n,a'}} \tag{6.1}$$

$p^*_{n,a}$ denotes the probability that σ^* plays action a at public history set n, N_{prior} is a Dirchlet prior distribution, and $c_{n,a}$ are the observed probabilities. Next the algorithm compute the posterior bucket probabilities, that are the probabilities the opponent is in each bucket at each public history set n.

Algorithm 2 Compute posterior bucket probabilities

for $b = 1$ **to** $|B_n|$ **do**
    $n' \leftarrow parent(n)$
    $a \leftarrow$ action taken to get from n' to n
    $\beta_{n,b} \leftarrow h_b \cdot s_{n',b,a}$
end for
Normalize the values β_n so they sum up to 1

where $\beta_{n,b}$ is the posterior probability that the opponent is in bucket b at public history set n, while h_b is the probability that *chance* makes the moves needed to put the opponent in bucket b, and $s_{n',b,a}$ denotes a probability model that the opponent plays a strategy leading to n' and once in state n' performs action a in bucket b. Finally a full model of opponent's strategy is computed considering the deviations between the opponent's posterior action probabilities and those of

the approximate equilibrium σ^* at n. The algorithm iterates over all public history sets and returns as output the best response of the opponent model.

6.1.1 Computing the opponent model

The approach used to compute the opponent model is to find the strategy that is closer to the equilibrium, because such a strategy will be less exploitable.

In the paper [2] researchers presented a custom weight-shifting algorithm that makes use of the approximated bucket ranking computed at each public history set from approximate equilibrium σ^*. Buckets are sorted by how often the opponent raise with them under σ^*. The algorithm shown by the researchers is designed for three actions. Opponent's strategy at n is initialized to the equilibrium σ^*, and the model γ_n of action probabilities values are set to those $p^*_{n,a}$ computed at equilibrium. Now suppose the opponent is taking action 3 more often than he should at n: this can be revealed comparing $\alpha_{n,3}$ to $\gamma_{n,3}$ and noting that the first is greater than the second one. The algorithm starts adding weight to the bucket $\hat{b}$ that plays action 3.

$$\gamma_{n,3} + \beta_{n,\hat{b}} \cdot (1 - \sigma_{n,\hat{b},3}) < \alpha_{n,3} \tag{6.2}$$

Next the increase in probability of playing action 3 is compensated by decreasing the probabilities of playing action 1 or 2.

Until the inequality 6.2 helds the algorithm continues to shift the probability mass in order to find an equilibrium between the opponent model probabilities and the posterior action probabilities.

Computing the opponent model and a best response is an heavy task, hence the procedure is designed to do so only every k repetitions.

Algorithm 3 Full algorithm: $DBBR(T, k, N_{prior})$

for $iter = 1$ **to** T **do**
  Play according to σ^*, the precomputed equilibrium strategy
end for
$opponent\_model = ComputeOppModel\ (N_{prior})$
$\sigma_{BR} = ComputeBestResponse$(opponent_model)
for $iter = T + 1$ **to** M **do**
  if $iter$ is a multiple of k **then**
    $opponent\_model = ComputeOppModel(N_{prior})$
    $\sigma_{BR} = ComputeBestResponse$(opponent_model)
  end if
Play according to σ_{BR}
end for

The game used for the experiments is two-player limit Texas Hold'em, a large scale game with 10^{18} states. Researchers ran the algorithm against several opponents. In particular it is worth to notice they tested the algorithm against the worst players in the 2008 AAAI computer poker competition: GUS2 and Dr. Sahbak. The choice is motivated by the fact the agent is designed to exploit weak opponents, while in case of strong ones a precomputed equilibrium strategy is to prefer.

GS5 is a bot used in the 2009 AAAI computer poker competition that plays an approximate-equilibrium strategy, computed using an abstraction with a branching factor of 15 in the first betting round, 40 in the second, 6 in the third and in the last one.

The parameters' values used in the test are $T = 1000$, $k = 50$, $N_{prior} = 5$. The reason of a so high value for T is because the algorithm needs to obtain at least a certain number of samples of the opponent's play before starting to exploit his game.

GS5 plays initially a approximate equilibrium strategy σ^* which is however precomputed using a smaller abstraction than its own: respectively 8, 12, 4, and 4 for the betting rounds. This was done for a matter of performance, and allowed to construct opponent models and best response in few seconds.

6.1.2 Experimental results

The developers compared four different algorithms, finding out that their custom weight-shifting DBBR outperforms the other three by a significant margin, especially against GUS2. The fact the algorithm

performs better against the AAAI competition's bots suggests that it may perform better in case of real opponents. The algorithm DBBR-L1 and DBBR-L2 are opponent modeling techniques that make use of a weighted distance: the first using norm_1 and the second using norm_2. Further details can be found on [2].

	GUS2	Dr. Sahbak
GS5	0.636 ± 0.004	0.665 ± 0.027
DBBR-WS	0.807 ± 0.011	1.156 ± 0.043
DBBR-L1	0.609 ± 0.054	1.153 ± 0.074
DBRR-L2	0.721 ± 0.050	1.027 ± 0.072

Table 6.1: Win rates in small-bets/hand for the bots listed in the row. The $\pm$ value is the standard deviation.

However while DBBR is effective to exploit weak opponents it might actually become exploitable in the case of strong ones. Hence an adaptive strategy should always be achieved in order to play equilibrium against strong opponents.

7 Computing the equilibrium following a Game-Theory approach

The approach described in this section considers the computation of equilibrium solutions using game theoretic principles.

The field of Game Theory provides analytical tools to study situations where multiple agents compete and interact within a particular environment [4]. Typically in these environments agents have a personal goal and objectives. A game involve a finite set of players and a finite set of moves that they can perform at specific points of the game. Once the game is finished a numerical payoff must be assigned to each player involved in the game. The payoff can be positive (gain), negative (loss) or zero (drawn). More over the game may include chance events and contain information that is not available to its players at the time they make the move.

7.1 Stochastic games

We can formally describe a stochastic game as a tuple (N, S, A, p, r) where:

- $N = 1, ..., n$ is a finite set of players.
- S is a finite set of states.
- $A(s) = (A_1(s), ..., A_n(s))$ is a tuple where for each player i and in each state $s \in S$, $A_i(s)$ is the set of available actions at state s for the player i.
- $p_{s,t}(a)$ is the probability to transit from state s to state t, taking the action $a \in A(s)$.
- $r(s) : S \to \Re^n$ is the payoff function, is a vector whose elements denotes the payoff for i-th player, at state $s \in S$.

At each state s the i-th player can choose a distribution of probability for his own action $A_i(s)$, at state s. If we denote the probability to select action a_i as $\sigma(a_i)$, we can define $\sigma(a)$ as the product of all the $\sigma(a_i)$ (*i.e.* $\sigma(a) = \prod_i \sigma(a_i)$).

Then it is possible to extend the definition of the transition function introducing the distribution probability.

$$p_{s,t}(\sigma) = \sum_{a \in A(s)} [\sigma(a) \cdot p_{s,t}(a)]$$

The goal of the agent is to maximize $\sum_{t=0}^{\infty} \gamma^t \cdot r_i(s_t)$, in which γ is the discount rate and its value belong to the interval $[0, 1]$.

A stochastic game with one agent is called *Markov Decision Process (MDP)*.

7.2 The algorithms for three-player tournaments

Recent works on the field suggest that it is optimal to adopt a jam/-fold preflop strategy at some point of the game, typically when the blinds are over a certain threshold. Several algorithms used in this context were developed by Sam Ganzfried and Tuomas Sandholm in their recent paper about computing equilibria in stochastic games of imperfect information [2].

In the following we do a brief review on their recent algorithm (and its variants) for computing an approximate jam/fold equilibrium in a

three-player tournament. All the algorithms can be logically divided into two parts: a *inner loop* and a *outer loop*. As regard VI-FP (Value Iteration - Fictitious Play), the inner loop uses an extension of smoothed fictitious play (FP) to determine ϵ-equilibrium strategies for all states (stack vectors), while the outer loop is an iteration that adjusts the values (VI) of the different states based on the new payoffs determined by the inner loop. Both of them are executed until no state's payoff changes by more than δ between outer loop iterations.

When a gamer plays a best a response in reply to his opponent strategies, we said he's acting **standard fictitious play**. In particular the smoothed version of fictitious play updates the strategy for player i based on a simple learning model:

$$s_i^t = \left(1 - \frac{1}{t}\right) s_i^{t-1} + \frac{1}{t} s_i'^t \tag{7.1}$$

where $s_i'^t$ is the best response of player i to the profile s_{-i}^{t-1}, which is the set of its opponent's strategies at the previous time-step $(t-1)$.

This algorithm was proven to converge to a Nash equilibrium in the case of two-players zero-sum games, but the same is not guaranteed for the case of multi-player or general-sum games.

Theorem 2 *Under fictitious play, if the empirical distributions over each player's choices converge, the strategy profile corresponding to the product of these distributions is a Nash equilibrium.*

States in poker tournaments are made of a vector containing the stack sizes for each player. Supposing three players in the game G, each state x is represented by a triple (x_1, x_2, x_3), where x_1 denotes the stack of the button, x_2 the small blind's stack, and x_3 the one belonging to big blind. When one of the stacks become zero, the corresponding player is eliminated from the tournament and the game becomes a two-player game, which can be solved with well-know techniques. Hence the states where one of the x_i is zero are ignored because they are managed in a different way.

ICM (Indipendent Chip Model) is a notorious heuristic used in the poker community that asserts a player's probability of winning is proportional to his fraction of all the chips, while a player's chance of placing second (third) is computed by the fraction of remaining chips conditioned on each other player coming in first (first or second).

The VI-FP algorithm first initializes the assignment V^0 of payoffs using the ICM heuristic for each player at each game state. Fictitious

Algorithm 4 *VI-FP*(γ, δ)

$V^0 = \text{initializeValues}()$
$\text{diff} = \infty$
$i = 0$
while $\text{diff} > \delta$ **do**
  $i = i + 1$
  $\text{regret} = \infty$
  $S = \text{initializeStrategies}()$
  while $\text{regret} > \gamma$ **do**
    $S = \text{fictPlay}()$
    $\text{regret} = \text{maxRegret}(S)$
  end while
  $V^i = \text{getNewValues}(V^{i-1}, S)$
  $\text{diff} = \text{maxDev}(V^i, V^{i-1})$
end while
return S

play is run using V^0 as payoff until it gets a $\gamma-$equilibrium. Then repeat the procedure using the payoff V^1 which results from the computed strategy. The algorithm halts when it reaches an iteration k such that each payoff of V^k differs from the corresponding payoff of V^{k-1} by less than δ.

During the experiments the researchers [2] found that both inner and outer loop of VI-FP converge, however there exist initializations for which the algorithm converges to a non-equilibrium profile. In order to verify the convergence they developed an *ex-post* checking procedure, that computes the best response of each player to the strategy profile S^*, computed with VI-FP. If the payoff improvement is less than ϵ for the players, then the original strategies constitute a ϵ-equilibrium.

Let's build the Markov Decision Process M induced by the strategy profile s^* computed with VI-FP.

Transitions are determined by probability distributions over the states. The rewards at each non-terminal state are null, while at each terminal state they are determined by the ICM payoff of the corresponding stack vector.

An optimal policy for M corresponds to a best response of each player in G to the profile s^*.

Theorem 3 *In our setting, if* v^0 *is initialized pessimistically (*i.e.*,*

$\forall s, v^0(s) \leq v^*(s)$*), value iteration converges (pointwise and monotonically) to* v^**.*

Policy iteration is an alternative way to find an optimal policy. The algorithm can be summarized in 4 steps:

1. Set $n = 0$ and initialize the policy π^0 so that it has non negative expected reward.

2. Let v^n be the solution to the system of equations

$$v(i) = r(i) + \sum_j p_{i,j}^{\pi^n} v(j)$$

   where $p_{i,j}^{\pi^n}$ is the probability of moving from state i to state j under the policy π^n. When multiple solutions are found, the minimal non-negative one is preferred.

3. For each state s with action space $A(s)$, set

$$\pi^{n+1}(s) \in \operatorname*{argmax}_{a \in A(s)} \sum_j p_{i,j}^a v^n(j)$$

   breaking ties so that $\pi^{n+1}(s) = \pi^n(s)$ whenever possible.

4. If $\pi^{n+1}(s) = \pi^n(s)$ for all s, stop and set $\pi^* = \pi^n$. Otherwise increment n by 1 and return to step 2.

For each state i in the MDP M, $v(i)$ is a variable corresponding to its value. Policy iteration aims to find the optimal policy π^*.

Since s^* is supposed to be a near optimal policy, it can be used as initial policy because it clearly produces non-negative expected total reward. Therefore the policy iteration algorithm is run on M (computed from strategy profile s^*) using as initial policy $\pi^0 = s^*$. Then ex post check returns the largest amount in expected utility any agent can improve by deviating from s^*.

PI-FP uses Policy Iteration as outer loop and Fictitious Play as inner loop. Hence PI-FP differs from VI-FP in how the values are updated during the outer loop. As result if the sequence of strategies $\{s^n\}$ determined by iterations of the Policy Iteration converges, then the final strategy profile s^* is an equilibrium. A nice property of PI-FP is that it can recover from a poor initialization because it uses the values resulting from evaluating the policy (not the one coming from initialization).

Algorithm 5 *PI-FP*(γ, δ)

$V^0 = \text{initializeValues}()$
$\text{diff} = \infty$
$i = 0$
while $\text{diff} > \delta$ **do**
  $i = i + 1$
  $\text{regret} = \infty$
  $S^0 = \text{initializeStrategies}()$
  while $\text{regret} > \gamma$ **do**
    $S^i = \text{fictPlay}()$
    $\text{regret} = \text{maxRegret}(S^i)$
  end while
  $M^i = \text{createTransitionMatrix}(S^i)$
  $V^i = \text{evaluatePolicy}(M^i)$
  $\text{diff} = \text{maxDev}(V^i, V^{i-1})$
end while
return S^i

FP-MDP is another new algorithm developed by the researcher, which reverse the roles of inner (Markov Decision Process) and outer loop (Fictitious Play).

As in the previous case policy iteration is preferred to value iteration because it gives a better start. Values are initialized using ICM and the initial strategy profile s^0 is generated. Then from this computed strategy profile, during the ex post check, a MDP is induced and constructed. After that a best response to each player using policy iteration is evaluated. Then the computed best response

Algorithm 6 *FP-MDP*

$S^0 = \text{initializeStrategies}()$
$i = 0$
while termination criterion not met **do**
$M^i = \text{constructMDP}(S^i)$
$S' = \text{solveMDP}(M^i)$
$S^{i+1} = \frac{i}{i+1} S^i + \frac{1}{i+1} S'$
$i = i + 1$
end while
return S^i

Algorithm 7 *FTPL-MDP*

$S^0 = \text{initializeStrategies}()$
$i = 0$
while termination criterion not met **do**
$\hat{S}^i = \text{randomPerturbation}(S^i)$
$M^i = \text{constructMDP}(\hat{S}^i)$
$S' = \text{solveMDP}(M^i)$
$S^{i+1} = \frac{i}{i+1} S^i + \frac{1}{i+1} S'$
$i = i + 1$
end while
return S^i

is combined with s^0 using fictitious play updating rule and s^1 is obtained. This last step is repeated until the sequence $\{s^n\}$ converges. The convergence is not guaranteed, but if it happens to occur, then the final strategy profile s^* is an equilibrium.

FTPL-MDP is a polynomial-time algorithm for regret minimization. The outer loop is performed using the algorithm FTPL (Follow The Perturbed Leader). The algorithm is similar to FP, except that instead of constructing directly the MDP from the computed strategy profile, this is built based on a random-noisy model of the strategy profile.

Experimental results on poker tournaments showed that the new algorithm *PI-FP* outperformed the old one *VI-FP*. The interesting thing was that the algorithms converged to an equilibrium despite the fact that they are not guaranteed to do so. The Figure 7.1 shows a comparison between the performance of the three algorithm VI-FP, PI-FP and FP-MDP. The x-axis represents the running time while the y-axis describes the maximal amount a player could gain by deviating from the current strategy profile in the outer loop. The y-axis can be seen as the ϵ of the equilibrium too. The graphic was generated running an ex post check over the strategies computed at the end of each outer loop iteration.

8 Near-Equilibrium Poker Agents

This section will examine some recent techniques used to compute a near Nash equilibrium, focusing on recent and at-state-of-the-art

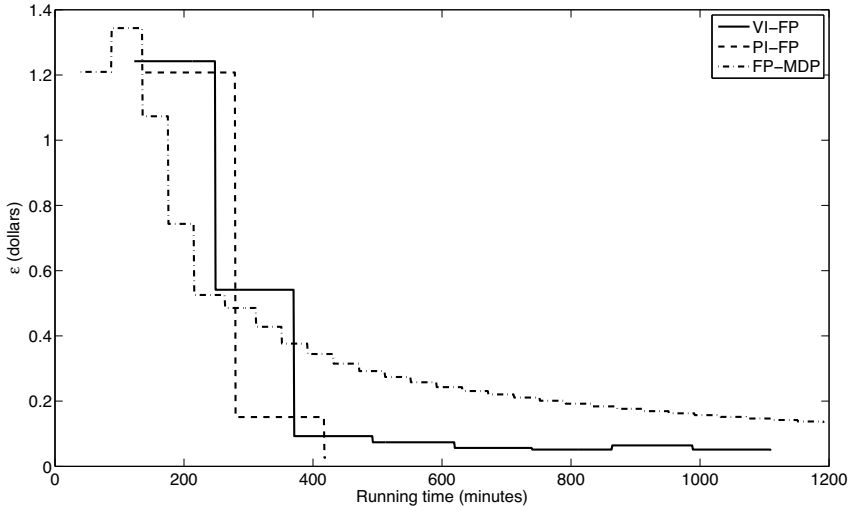


Figure 7.1: Running time required to find an ϵ equilibrium for VI-FP, PI-FP and FP-MDP algorithms.

algorithms. In all the cases the agents used abstractions. *Aggrobot* is one of them, with peculiarity in being an agent that generates separate preflop and postflop models that are then combined to produce a near-equilibrium strategy for the game of no-limit hold'em. Aggrobot uses a betting abstraction that defines four different and separate betting actions: half the pot, the exact pot's amount, 2 times the pot, and all-in (all the stack). Aggrobot was only able to reach a near equilibrium strategy but it was a good result.

Next, we analyze a set of iterative algorithms typically used to compute an approximated Nash equilibrium, also called ϵ-equilibrium, that have fewer memory requirements and hence can solve larger models than LP.

8.1 Fictitious Play and Range of Skill

These two techniques have been used to produce solid limit hold'em agents. Fictitious Play is centered around the idea that the strategies of two players repeatedly playing a game against each other could improve and adapt over time. The algorithm starts with the players adopting casual strategies. Then random game situations are proposed to the players that are therefore involved in a training phase.

Hence the players can determine the correct strategy based on their opponents' strategy. As the number of iteration increases the computed strategies approach a Nash equilibrium. It's a sort of learning.

The second one, *Range of Skill*, is a iterative procedure that considers creating a sequence of agents, where the next agent created employs a strategy that can beat the previously created agent by at least an amount of ϵ. As the number of agents in the sequence increases the agent's strategies approach an ϵ-Nash equilibrium. The algorithm calls repeatedly a procedure known as *generalized best response* that computes the best response to the considered subset of the allowed strategies (restricted game). The algorithm works by returning equilibrium strategies for restricted games of increasing size. As the number of iterations increases the returned strategies approach the approximated ϵ-Nash equilibrium.

A state-of-the-art procedure is the Excessive Gap Technique (EGT) which is an iterative algorithm that requires approximately $O(1/\epsilon)$ iterations to approach a ϵ-Nash equilibrium. As the number of iterations increases the ϵ value is consequently reduced.

In the case of two-player zero-sum games the first player tries to maximize its payoff and the second player attempts to minimize first player's payoff.

The use of EGT resulted in one of the first agents able to solve a model with four final rounds of play. Recently Sandholm and Gilpin presented two improovements for EGT algorithm, one based on randomized sampling while the second using a cache-coerent non-uniform memory access architecture.

8.2 CFR for computing ϵ-Nash equilibrium

CFR, which stands for Counter-Factual Regret minimization, is a iterative recent technique used to compute strategies that converge to an ϵ-Nash equilibrium, with proof. The algorithm relies on iteratively minimizing a counterfactual regret value. A major advantage of this algorithm is that it requires memory linear in the size of informations sets.

As the number of iterations goes to infinity the overall average regret approach zero.

CFR does not minimize one overall regret value only, but it separates regret values into different information sets and minimizes their individual regret values. The counterfactual aspect regards the fact

that calculations are weighted by the probability of reaching a particular information set.

CFR algorithm is confirmed to be theoretically valid for two-player zero-sum perfect recall games. However a recent research in the field [5], has brought Nick Abou Risk and Duane Szafron to the belief that CFR would perform well in multiplayer games as well.

To test this technique they created several 3-player limit Texas Hold'em poker agents, whose number of game states is about 10^{24}. The goal was to compute winning strategies for these large extensive multiplayer games.

However CFR is only guaranteed to converge to an ϵ-Nash equilibrium for two-player zero-sum perfect recall games, and even if CFR succeeds in finding an ϵ-Nash equilibrium strategy, we can't affirm without any doubt that it will perform well in a multiplayer game.

The difference between the highest utility achievable and the utility of the action that was taken is called **regret**. CFR algorithm minimizes the positive immediate counterfactual regret at each information set, which was demonstrated to lead to a overall regret minimization. Therefore this act of minimization conducts to a ϵ-Nash equilibrium profile.

In addition CFR may compute an abstract game best response against a static opponent. The algorithm is flexible enough, and is quite robust in generating strong strategy profiles for two-players games where only one between perfect recall and zero-sum constraints subsists.

As a matter of fact the research [5] is the first one reporting results about applying CFR to large multiplayer games.

Since the computation of the ϵ-Nash equilibrium using CFR takes days or even weeks of computation time, researchers evaluated the new algorithms using a simplified version of poker with constraints on exhibiting properties such as stochasticity, hidden information, or action-based utility.

8.3 Kuhn and Leduc Hold'em

Kuhn is a poker game invented in 1950, in which we can find important properties such as bluffing, inducing bluffs and value betting. The three player variant used for the experiment has a tree with 25 game's nodes and a depth of 6. In Kuhn poker the deck has only 4 cards of the same suit: $K > Q > J > T$. Each player is dealt one private card and antes 1 chip before the cards are dealt. There is

one betting round with a 1-bet cap. If there is no outstanding bet, a player can check or bet 1 chip, while if there is one a player can fold or call.

Leduc Holde'm is closer to real poker games than Kuhn, since it includes aspects such as multi-round play, community cards, rounds with different bet size, split pots and raising that Kuhn poker indeed ignores. Leduc is originally born as heads-up variant, however it was extended to three player for the purpose of testing CFR.

In Leduc Holde'm the deck consists of 4 ranks and 2 suits (*e.g.* T♡, T♢, J♡, J♢, Q♡, Q♢, K♡, K♢), using the well-known ranking $K > Q > J > T$. Each player receives one private card and antes 1 chip on the pot before the cards are dealt. A upper limit of 2 bets is imposed and two betting rounds with different bet size take place: preflop (2 chips) and flop (4 chips). One community card is faced up before the flop betting. After that the active players showdown. As regard hands ranking, a paired hand beats an unpaired hand. And if there are no pairs the high card wins.

In case of two players tying with the same highest card, the pot is split between the players.

8.4 Evaluating 3-player CFR strategies

Consider a 3-player strategy profile $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ computed using CFR. We identify as σ_{-i} the set of all strategies in σ except σ_i.

With this notation in mind we can again express the well-known concept that if σ is a ϵ-Nash equilibrium strategy profile then no player i can gain more than ϵ by deviating from σ_i while σ_{-i} remains fixed.

Therefore CFR can generate a best response σ^{BR} to the strategy profile σ. The best response for each player is obtained by fixing σ_{-i}

After the strategy profile is computed, a method to verify whether it is an ϵ-Nash equilibrium or not, consists in computing the utilities for each position of the strategy profile σ by playing three σ's against each other and compute the utilities of the best response in each position by playing one σ^{BR} against two σ's. Then comparing σ^{BR} utilities in each position to σ's utilities, how much extra best response wins in each position should be determined in order to see the magnitude of improvement. If best response does not improve by more than ϵ (as mean of the over all positions) then σ is an ϵ-Nash equilibrium.

Results showed that approximated equilibrium was found only for 3-player Kuhn game, while Leduc Hold'em did not converge. Hence

the hope of usign CFR to generate ϵ-Nash equilibrium for more complex games has failed. A best response provides a metric for how much an agent could improve its utility. CFR is able to compute the best game for an abstracted heads-up limit hold'em. However computing it for a multiplayer version is not realistic since it would take months.

To bypass the problem the authors of [5] designed a set of benchmark agents and made them play millions of hands between each other in order to evaluate those created using CFR.

8.5 Benchmark agents vs CFR agents

Among the benchmark agents used there was Poki, the winner of the multiplayer limit event of the 2008 CP Competition. Poki uses a collection of technologies including an expert system preflop, and Monte Carlo simulations designed to estimate the most remunerative action to take against an opponent model.

Chump agents such as *Always-Fold*, *Always-Call*, *Always-Raise* with the obvious built-in behavior, were employed for testing purpose together with *Probe*, a chump agent that chooses with equal probability between call and raise actions.

Three agents were constructed using CFR: one with perfect recall, the other two with imperfect recall.

Just think that it takes over 32 GB of RAM to store the game tree of a three player limit Hold'em, with no betting abstraction and only two buckets per round.

Researchers constructed PR2, the agent, using a 2-bucket *perfect recall* abstraction for 20 million iterations on 32 GB machine. However the betting upper limit on the river was reduced from 4 to 3, because of memory limitation. In the event PR2 falls off-tree while playing, it simply calls. It took about three weeks to complete the 20 million iterations.

Using a new machine with 64 GB of RAM researchers were able to create a 16-bucket *imperfect recall* abstraction. They used CFR on this abstraction creating 2 agents: IR16S, computed using 20 million iterations and IR16L, originated by 43 million iterations. The latest CFR computation took about one month.

Agent strength may be evaluated in *bankroll events*, in which players are ranked based on their sole win rate, or in *elimination events*, where the winner is determined by recursively removing the worst agent evaluated using a financial metric as in bankroll events.

The recursion ends when three agents only remain in game. The three players are ultimately graduated considering the win rate as in bankroll.

As experiments a tournament was run with the lately exposed seven 3-player limit Hold'em agents: *Always-Call*, *Always-Raise*, *Probe*, *Poki*, *PR2 (Perfect Recall 2)*, *IR16S (Imperfect Recall 16 Small)* and *IR16L (Imperfect Recall 16 Large)*.

The tournament consisted of 35 matchups where each player played against 15 pair of opponents. Each matchup being 1.2 million hands.

PR2	IR16S	IR16L	Poki	Always-Raise	Probe	Always-Call
530	341	333	327	-461	-462	-607

Table 8.1: Overall win rates for a 7-agent 3-player tournament, expressed in millibets/hand, with 95% of confidence.

All three of the CFR-generated agents developed in [5] outperform Poki, the 2008 CPC multiplayer champion. PR2 is the best exploitative agent, since it obtained a value of 1086 mb/hand against *Always-Call* and *Always-Raise*, and 734 mb/hand against *Always-Raise* and *Probe*. On the contrary *IR16L* and *IR16S* earned their worst result against these latest two. This is an exception since by in the other matchups it is clear to deduce that *IR16* agents are the least exploited ones.

8.6 Introducing Heads-Up Experts

Because heads-up situations arise very commonly in 3-player limit Hold'em, it is good practice to adapt the algorithm to this condition. In fact after one player folds, the game becomes two-player, zero-sum and perfect recall. Keeping this in mind we can deduce that it is guaranteed to find a ϵ-Nash equilibrium. Moreover the heads-up subtree is smaller and larger abstractions can be performed.

In order to locate the point of the game in which the head-up emerges, frequencies betting sequences that bring to a fold were computed from 1.2 million hands of self-play between three identical agents. The most frequent ones are shown in Table 8.2.

The idea proposed by [5] is to use a static strategy such as *IR16L* and switch to a Heads-Up Expert (HUE) when one of the sequences of

Sequence	Poki	PR2	IR16S	IR16L	Button	SB	BB
f	20.67	29.27	36.03	36.14	1-2	-	-
rf	9.55	25.60	26.40	26.37	4-5	1-4	-
cf	16.43	0.02	0.09	0.04	3	1-2	-
rrf	0.54	1.51	6.64	6.61	3-5	5	1-4
rcf	1.52	9.96	0.66	0.65	3-5	4	1-2
crf	3.35	0.01	0.04	0.02	3	5	1-2
total	***52.05***	***66.38***	***69.84***				

Table 8.2: Frequencies of the most common bet sequences ending in a fold, between 3 identical agents using 4 different strategies for 1.2 million hands of self-play.

Table 8.2 occur. The main problem remains how to knit the strategies together. We need to determine a reasonable strategy to *seed* the HUE subtrees. Uniform and expert seeding are two strategies employed. In particular the expert one makes use of preflop strategy based on a expert system, such as advise coming from poker professionals or strong computer poker agents. Hands are placed in buckets with ranking $5 > 4 > 3 > 2 > 1$. For example, as we can see in Table 8.2 in sequence rcf, Button raises with hands in 3-5, Small Blind calls with hands in bucket 4 and Big Blind folds with hands in 1-2.

Finally the authors of [5] conducted a second 7-agent 3-player tournament without including any chumps. This was done to simulate a more realistic competition. The overall results are shown in Table 8.3.

IR16L	Ex16	Un16	ExM2	UnM2	PR2	Poki
57	49	26	8	-21	-47	-72

Table 8.3: Overall win rates for a 7-agent 3-player tournament with 4 HUEs, expressed in millibets/hand, with 95% of confidence.

The new agents introduced are particular cases of the old ones using uniform or expert seeding of Heads-Up Experts.

The top 2 bankroll players were *IR16* and *Exp16*, winning in every matchup they participated. The worst were Poki and PR2. The confidence of the overall winrate is ± 6 mb/hand as in Table 8.1.

UnM2	is *PR2* using uniform seeded HUEs.
ExM2	is *PR2* using expert seeded HUEs.
Un16	is *IR16L* using uniform seeded HUEs
Ex16	is *IR16L* using expert seeded HUEs

Researchers demonstrated how effective CFR can be for generating very abstract, imperfect recall, 3-player agents, regardless the fact no theoretical guarantee exists. CFR can in fact be used to generate two-player ϵ-Nash equilibrium profiles, called heads-up experts (HUE), when one among the three players folds during preflop phase.

9 Human Behavior on Simplified Poker Game

Bluffing and betting behavior are a very important aspect that should be taken into account when players have to decide which strategy to employ at some point of the game. In their recent paper [3], Darryl A. Seale and Steven E. Phelan proposed an experiment based on a pure strategy simplified poker (PSP) game, where 120 human players played against a computer programmed to play either the equilibrium solution or a fictitious play learning algorithm specifically designed to take advantage of weak opponents.

The interesting result in this research was that human players made a considerable amount of errors by betting when they should have checked or by calling when they should have folded.

9.1 Experiments in the past

First in 1938 Borel, and then in 1947 Von Neumann and Morgenstern performed experiments regarding the game of poker.

Von Neumann and Morgenstern derived an analytical solution for a simplified poker game that required players to draw a continuous value from a deck in the range $[0, 1]$. The solution required to make a bet on highest and lowest values and make a check on intermediate ones.

Borel proposed and solved a two-player poker game that requires the players to submit an ante of one unit before the dealing of the cards (a random number drawn from the interval $[0, 1]$). After the cards are dealt the first player, denoted as P1, can bet or fold. If P1

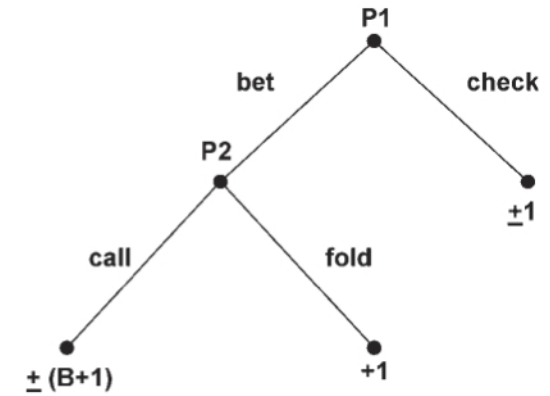


Figure 9.1: Decision tree for the Von Neumann poker game

folds the second player, P2 wins the pot, otherwise P2 has the option of calling or folding. The solution to the Borel game confirms the presence of a single value threshold based on which the decision of betting or not is taken by P1.

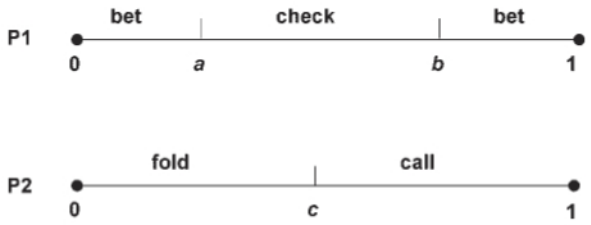


Figure 9.2: Bet, check, call and fold zones for P1 and P2 in the Von Neumann poker game

The two-players game proposed by Von Neuman and Morgenstern differs from the one of Borel by the fact that P1 chooses between bet or check (not fold), hence he's forced to play. Furthermore if P1 checks P2 can't take a decision and both the hands are faced up and the highest hand wins the pot. However if P1 bets, P2 can decide between folding or calling. The optimal policy as already introduced, requires P1 to make a bet with hands valued less than a threshold a and with hands valued more than a threshold b. On the contrary P1 must check if its own hand's value is between a and b, with $b > a$. The solution for P2 says that only two decision zones should be established: one for folding, the other one for calling.

9.2 PSP: a pure-strategy simplified poker game

The game employed in the experiment differs from the Von Neumann game by the fact that hands are dealt without replacement from a deck containing only 7 ordered cards: $\{2, 3, 4, 5, 6, 7, 8\}$.. The ante is of one unit and the bet is fixed to two units.

After the cards are dealt P1 can check or bet. If P1 checks, then both the cards of P1 and P2 are faced up. If P1 decides to bet (2 chips), P2 can then fold or call. If P2 folds P1 wins the pot, while if P2 calls by adding 2 additional chips in the pot, the cards are faced up. The pot goes to the player with the highest card.

Optimal solution for P1 is to bluff with card $\{2\}$, check with $\{3, 4, 5, 6\}$ and bet with $\{7, 8\}$. On the other hand for P2 the optimal strategy is to fold with hands $\{2, 3, 4, 5\}$ and call with $\{6, 7, 8\}$. Following the optimal play the game favors P1.

120 subjects, with approximately same percentage of males and females, played this Pure strategy Simplified Poker (PSP) game against the computer, programmed to play either the fictitious play (FP) or the equilibrium solution (ES). Subjects were motivated by a potential cash award of $25.

FP was specifically designed to take advantage of (exploit) the weak opponents. It is the same iterative process designed by Brown in 1951 for finding approximate equilibrium solution for discrete, two-player, zero-sum strategic games. This particular technique maintains an history on the relative frequency distribution of opponent's choices and based on that computes the selection of the best reply strategy.

Behavioral Learning Theory (BLT) researchers believes that subjects use different heuristics (or *mental models*) in order to solve the task. The choice of this *mental model* is influenced by the information available to the player about the game. For example by providing information on the opponent's threshold we can dramatically increase the amount of bluff during a game.

The art of bluffing, betting on low cards, may not form part of a typical subject's behavioral model and must instead be learned.

Behavioral Learning Theory refers to the optimization of play in a selected model while Cognitive Learning Theory deals with the selection of the right model. It is interesting to study how the optimal strategy can be calculated in PSP game, in particular to understand the interplay between cognitive and behavioral learning.

With respect to several previous experiment in the field performed by other researchers, PSP used only seven hands, thus the number of

world' states to learn was reduced.

The computer played ES (equilibrium solution) against one half of the subjects and FP against the other half. According to BLT subjects playing against FP should have a faster learning rate than those playing against ES. Whereas according to CLT those to have a faster learning rate are the subjects assigned to P2 role, regardless the condition. In fact subjects were randomly assigned to 4 conditions:

P1-FP The subject plays the role of P1 and computer acts with fictitious play in the role of P2.

P1-ES The subject plays the role of P1 and computer acts with equilibrium strategy in the role of P2.

FP-P2 The subject plays the role of P2 and computer acts with fictitious play in the role of P1.

ES-P2 The subject plays the role of P2 and computer acts with equilibrium strategy in the role of P1.

Although players knew their role (P1 or P2) in the game, they did not know the strategy employed by the computer. They also ignore that the game would last 200 trials.

9.3 Experimental results

One first observation was about examining the decision behavior of subjects assigned to the role of P1. The results, plotted in Figure 9.3, show that bluffing by subjects in the role of P1 in condition P1-ES is consistent with equilibrium solution in 52% of the times, while the consistency decrease to 35% in the case P1 plays in condition P1-FP. In both the conditions the behavior that exhibit more consistency with equilibrium predictions was the Bet, with 90% in P1-ES condition and 94% in P1-FP.

As regard the analysis of folding/calling behavior, in the case of subjects covering the role of P2, the calling action was the one employed more accurately (near the equilibrium). On the contrary P2 chose folding action half times it shouldn't have based on equilibrium solution.

Using some techniques researchers understood that subject's decision behavior didn't change (adapt) with experience in the PSP game. In other words subjects failed to learn or approach an equilibrium play with repetitions of the game. This may also be influenced

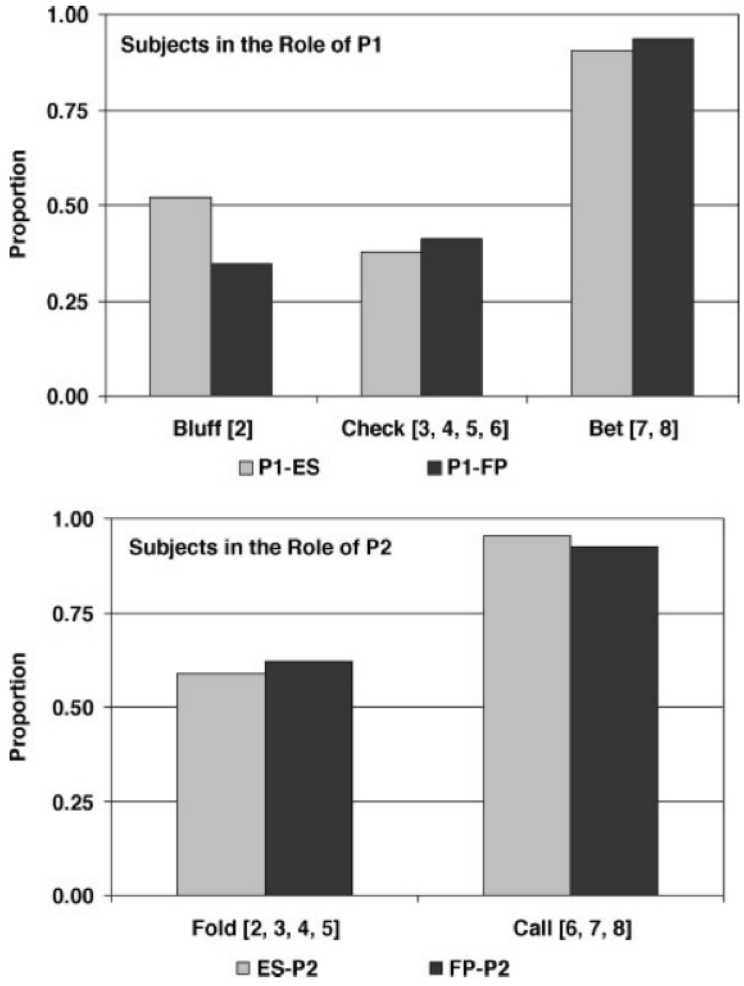


Figure 9.3: Proportion of decisions consistent with equilibrium predictions by type of decision and condition

by the low number of iterations (only 200) the subjects were involved in.

Probability matching behavior states that the likelihood of betting should monotonically increase based on the value of the hand. Results shown in figure 9.4 confirm this response with exception for the card 2. The non-monotonic result for this card is explained by the fact that several subjects understood the necessity to bluff when handling the lowest card (hoping a fold reply by the computer), in the intent of maximizing the utility.

As regards exploitability the experiment underlined that subjects,

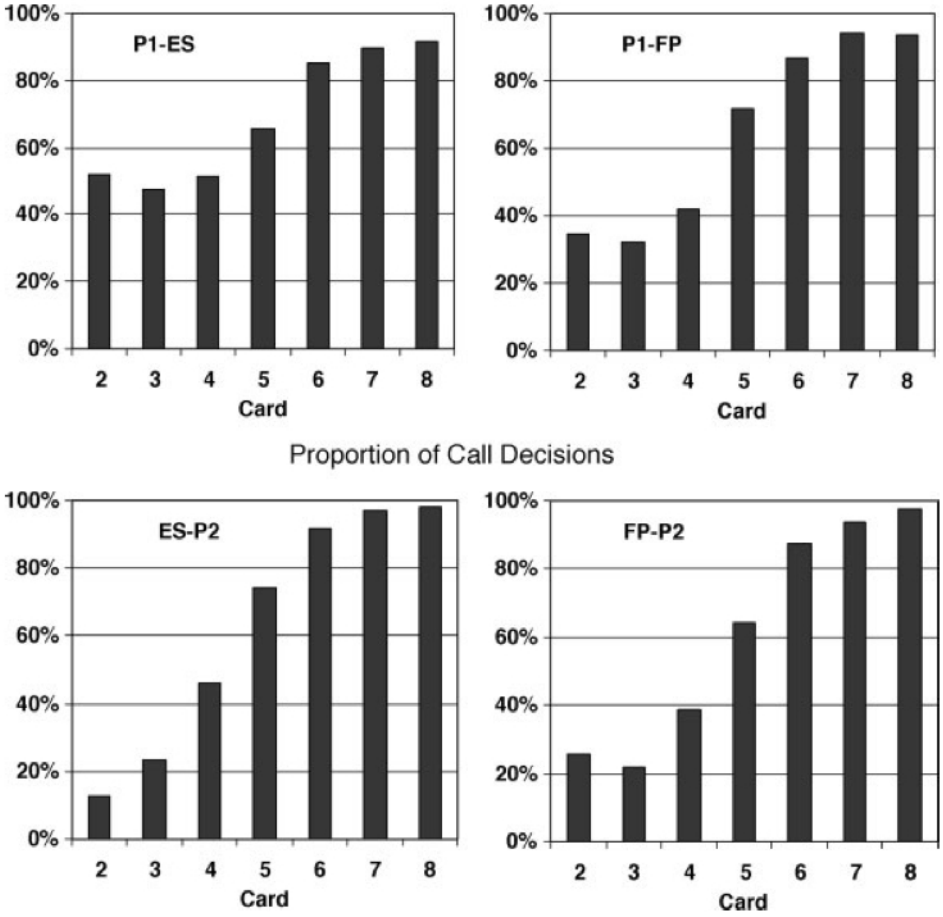


Figure 9.4: Proportion of bet or call decisions by condition and card value

in the role of both P1 and P2, made a considerable number of non-equilibrium decisions that didn't decrease during the trials. Several individual differences were observed as predicted by Cognitive Learning Theory, which provides as justification the heterogeneity of the *mental models*. No consistent pattern among subjects emerged. Researchers examined individual strategy profiles noting that many of these profiles tended to support the probability matching behavior. However they also observed strategy profiles that were somewhat irrational or hard to classify. After the game, some subjects, when asked about the strategies they employed, reported they were deliberately playing irrationally in order to fool their computer opponent. However the computer opponent ignored the irrational play when in

ES mode, and exploited it when in FP mode.

In conclusion subjects decisions were far away from the optimal solution and did not improve during the 200 repetitions of the game. Most of the errors for the subjects assigned to the role of P1 were due to the fact they didn't bluff with the lowest card, and caused by an incorrect choice of bet action on intermediate-valued cards when a check should instead be preferred. As regards subjects playing the role of P2, most of the errors were due to calling on intermediate-valued cards when they instead should have folded.

As in previous studies players failed to bluff in the role of P1 and called to often in the role of P2. Finally these results provide little support for either BLT or CLT. Different aspects, such as the BLT's belief about subjects learning faster in FP conditions or the CLT's prediction that subjects assigned to the role of P2 would learn faster, were not confirmed and supported by the results.

The study suggest that games with counter-intuitive elements (such as the bluffing behavior) might take longer to learn than predicted, because the subjects need to adapt their *mental models* to include these counter-intuitive elements in their solution sets.

10 Conclusions

The vast topic of computer poker game has been investigated, putting an emphasis on the game theoretic approaches, such as the approximated Nash equilibrium, employed to find a solution to the game. In particular recent techniques such as Counterfactual Regret Minimization have been analyzed in detail. In first place Texas Hold'em, the most popular poker game of our times, has been described with its rules and its background. Decision factors influencing human player, such as the hole cards, or the number of players, have been analyzed, without neglecting very important decision factors such as the opponent' strategies, and making a classification of them.

In the second section the extensive form game have been formalized, introducing the concept of strategy as probability distribution over the possible actions and describing concepts such as Nash Equilibrium or best response.

Then the necessity to introduce abstractions in order to find approximated Nash equilibria has been fully inspected, illustrating the various possible ways of abstraction, from the very basic card abstraction, to the more radical betting round elimination. About the

abstraction topic, novel algorithms coming from recent research [1] have been described.

Furthermore the possibility of exploiting the opponent's weakness through a technique that involves the modelling of the opponent, has been analyzed in detail, in parallel with recent works such as [6].

In the next section we described the computation of approximated Nash equilibrium using all the already explained presumptions such as abstractions or opponent modeling and exploitation. Recent algorithms involving three-player tournaments in stochastic games of imperfect information described in [2], and the well-known technique of Counterfactual Regret (CFR) Minimization [5] have been discussed, including the useful heads-up expertise at some point of a three-player game (i.e. when one of the three player folds).

In the end of the paper we introduced another point of view of the problem, showing a recent research published in the well-know *Journal of Behavioral Decision Making* [3]. In this research results of an experiment involving 120 subjects playing against a programmed computer pure-strategy simplified poker game are explicated, showing in particular the influence of theories such as behavioral (BLT), cognitive learning (CLT) or probability matching behavior on the topic of poker game.

References

[1] John Hawkin, Robert Holte, Duane Szafron, *Automated Action Abstraction of Imperfect Information Extensive-Form Games*, Department of Computing Science, University of Alberta, 2011

[2] Sam Ganzfried, Tuomas Sandholm, *Computing Equilibria in Multiplayer Stochastic Games of Imperfect Information*, Department of Computer Science, Carnegie Mellon University, 2009

[3] Darryl A. Seale, Steven E. Phelan, *Bluffing and Betting Behavior in a Simplified Poker Game*, Journal of Behavioral Decision Making, University of Nevada Las Vegas, 2009

[4] Jonathan Rubin, Ian Watson, *Computer Poker: A Review*, Department of Computer Science, University of Auckland, 2011

[5] Nick Abou Risk, Duane Szafron, *Using Counterfactual Regret Minimization to Create Competitive Multiplayer Poker Agents*, Department of Computer Science, University of Alberta, 2010

[6] Sam Ganzfried, Tuomas Sandholm, *Game Theory-Based Opponent Modeling in Large Imperfect-Information Games*, Computer Science Department, Carnegie Mellon University, 2011

[7] Andrea Cannizzaro, *Il Texas hold'em in4e4'otto: regole, strategie e consigli utili per vincere live e online*, Edizioni L'Airone, 2010

[8] Wikipedia, *http://www.wikipedia.org/*