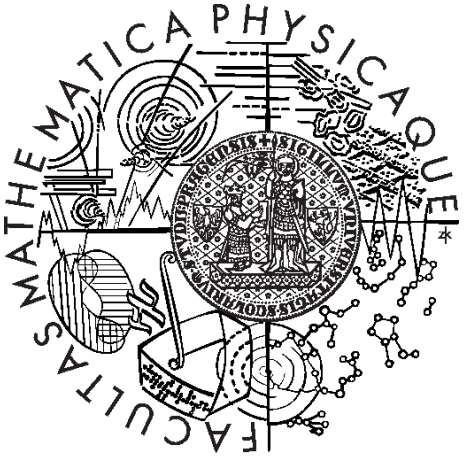
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Martin Schmid

Game Theory and Poker

Department of Applied Mathematics

Supervisor of the master thesis: Milan Hladík
Study programme: Informatics
Specialization: Discrete models and algorithms

Prague 2013

I'm very thankful to my supervisor, Mgr. Milan Hladík, Ph.D. He supported me and provided helpful comments. He was also very understanding as for the homework duties for his classes, especially when me and my roommate were working late hours to make our agent for the competition before the submission deadline.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Teorie her a poker

Autor: Martin Schmid

Katedra: Katedra aplikované matematiky

Vedoucí diplomové práce: Mgr. Milan Hladík, Ph.D.

Abstrakt:

Tato práce představí základní koncepty teorie her. Jsou představeny nezbytné modely a koncepty, následovány výpočetní složitostí odpovídajích algoritmů. Poker je formalizován v rámci modelů teorie her. Nejnovější algoritmy pro tento model her jsou vysvětleny pomocí aplikace na poker. Práce také podává přehled o tom jak mezi sebou mohou jednotlivé programy soutěžit, konkrétně na příkladu Annual Computer Poker Competition a příhlášených programů. Nakonec je představen nový výsledek týkající se extensive form her s mnoha akcemi.

Klíčová slova: Teorie her, poker, Nash equilibrium, hry s neúplnou informací

Title: Game Theory and Poker

Author: Martin Schmid

Department: Department of Applied Mathematics

Supervisor: Mgr. Milan Hladík, Ph.D.

Abstract: This thesis introduces the basic concepts of the game theory. Necessary models and solution concepts are described. Follows the summary of the computational complexity of these concepts and corresponding algorithms. Poker is formalized as one of the game theory game models. State of the art algorithms for the extensive form games are explained with the application to the Poker. The thesis also introduces the Annual Computer Poker Competition and participating programs. Finally, new result about the extensive form games with many actions is presented.

Keywords: Game theory, Poker, Nash equilibrium, Extensive form games

Contents

Introduction

Poker is an interesting research topic due to many reasons.

Chess is similar to Poker in a sense that due to the size of the game, we can't solve the entire game. Even though we can't solve the game of chess, state-of-the art chess programs win against the top human players. That being said, can we apply similar algorithms and create winning poker agents? The answer is no. Algorithms being used for games like *chess*, *go* etc. can't be used here. The reason is that the Poker is a game with imperfect information (see chapter 2 for details, informally - player can't see opponent's cards). We need different algorithms to solve games with imperfect information.

First, note that the existence of optimal strategy in Poker is far less obvious than the existence of optimal strategy in Chess. As we will see, the optimal strategy indeed exists. There are algorithms that can solve games with imperfect information, but the bottleneck is the game size. The study of Poker has led to new, state-of-the art algorithms able to solve games orders of magnitude larger than ever before, such as *counterfactual regret minimization* [44] and *EGT for extensive form games* [15] (we will see the first algorithm in chapter 7).

Using these new algorithms, current agents beat top human Poker players in no-limit version of Hold'em Poker [9]. Poker research community agree that computers will beat humans in no-limit version within 2 or 3 years.

Note that the theoretical and algorithmical results are not limited to Poker. It's just the case that Poker has served as the test bet due to the imperfect information and the size of the game.

First chapters introduce the reader to the game theory. Some general ideas are presented and few game theory models are defined. Chapter 3 introduces the concept of Nash equilibrium and briefly talks about the existence of such strategies. Following chapter further develops this concept and shows the computability results.

Chapter 5 finally shows the game of Poker, Poker rules and different game types. Annual Computer Poker Competition is also presented. Chapter 6 deals with Poker minigames that are easily solvable. Chapter 7 introduces the concept of *regret*, *counterfactual regret* and shows *regret* minimization algorithms. Chapter 8 deals with game abstractions and applies regret minimization algorithms to the abstracted game of Poker. Chapter 9 shows new result about no-limit betting in Poker (this result is not limited to Poker, it applies to two-players extensive form game with imperfect information).

1. Game theory

1.1 Introduction

Game theory can be defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers [26]. The models of game theory are highly abstract representations of classes of real life situations [29].

Some situations are clearly "games" and one can imagine that some math can be used to solve them - chess, tic-tac-toe, poker etc. But game theory is applied in many surprising situations - traffic in big cities [28], stability of price systems in economy [29] and even evolution of lifeforms in biology [38] (Evolutionary game theory).

To solve any game, we need to create or apply an appropriate mathematical model for our situation (we need to model participants, decisions, strategies, outcomes etc). Then we need to "solve" this model - compute the best strategies for all participants (or for some of them).

I briefly describe some concepts without any formal definitions. Later chapters define all of this formally.

1.1.1 Models

Basically, any game theory model should define:

- game players
- what are the possible actions player may take
- what will happen when all players choose some actions (*consequences*)
- player preferences on the possible consequences

There are many different game theory models, and they apply to different real-life situations. For example, one model can be used to solve rock-paper-scissors and different one to find a best price for some product. Later, I will describe several models and mention some games those models apply to. Even though first game theory models were motivated by real-life applications [1], there are many models and results that are purely theoretical.

1.1.2 Strategies

Player's possible actions are usually defined using a finite or infinite set. For example, in rock-paper-scissors the set would be finite with only 3 members. On the other hand, in some bidding games where a player may bid any amount of money, the set would be infinite. Player's strategy refers to action (or sequence of actions) he decided to play whenever he plays the game. Sometimes, this may be too restrictive (playing rock all the time may not be a great strategy). These strategies are called *pure strategies*. If we allow player to choose his strategy randomly, his strategy now becomes a probability distribution over his pure strategies. This is called *mixed strategy* (clearly *pure strategies* $\subseteq$ *mixed strategies*)

1.1.3 Optimal play

When playing any game, player usually wants to "win". To do this, player should play as good as possible. But what does it mean to play "good" strategy? Let's first define what does it mean to play "badly":

Player played badly, if he could gain higher value by choosing different strategy (given the other players stick to their's previous strategies)

Similarly, by playing "good" strategy we mean:

Player played good, if he could gain higher value by choosing different strategy (given the other players stick to their's previous strategies

Now imagine that all players want to play as good as possible - all players want to choose some *good* strategy. If there's such a solution, for which no player can improve by choosing different strategy, all players could "agree" on this solution. This solution may or may not exist, but if there is one, it's usually referred to as the *steady state* or *Nash equilibrium* and it's very important concept in game theory.

Later sections describe all of this formally and show what are the conditions for the existence of the *steady state*. Chapter 4 focuses on the computability of such solutions.

2. Game theory models

In this chapter, I describe few models and show some games these models can be applied to.

2.1 Strategic games

A strategic game is a model in which each player chooses his strategy and then all players play simultaneously.

Before any formal definitions, let me show some simple games that are simultaneous and thus can be modeled using strategic games.

Rock, paper, scissors Very famous game where two players simultaneously select either rock, paper or scissors [35]. Player either wins, looses or draws.

Rock, paper, scissors, lizard, spock Advanced version of the previous game. [24]

Prisoner's dilemma Two prisoners are being interrogated. Prisoner can either stay quiet or cooperate. If both stay quiet, they both get 2 years. If they both confess, they get 6 years. But if only one cooperates, he is offered a bargain and is freed, but the other prisoner gets 10 years

Battle of sexes Boy and a girl decide where to go on a date. There's an important football match and some opera tonight. Let's suppose that happinnes can be represented by an integer (and sadness by negative one).

- Both choose to go to football

  Boy is happy (10) and girl is quite sad (-6)

- Both choose to go to opera

  Boy is sad (-8) and girl is happy (10)

- Boy chooses football and girl opera

  They argue and are both sad (-20).

- Boy chooses opera and girl football

  Well, that could never happen, so arbitrary number is fine.

Definition 2.1.1. A strategic game $\langle N, (A_i), (u_i) \rangle$ [29] consists of

- a finite set N (the set of players)
- for each player $i \in N$ a nonempty set A_i (the set of actions available to player i)
- for each player $i \in N$ a preference relation $\succeq_i$ on $A = \times_{j \in N} A_j$ (the preference relation of player i)

If the set A_i of actions of every player is finite, then the game is *finite*. For a finite strategic game (and under some conditions also for infinite games), a preference relation may be conveniently represented by a **payoff function** (**utility function**):

$$u_i : A \to \mathbb{R} \tag{2.1}$$

Value of this function is a payoff(utility) for a given player, that is - how much the player wins or looses. From now on, I will refer only to finite strategic games.

Definition 2.1.2. Zero sum games are games for which:

$$\sum_{i \in N} u_i(a) = 0 \ \ \forall a \in A \tag{2.2}$$

In other words, utility gained by one player is utility lost by the others.

2.1.1 Strategies

Definition 2.1.3. Pure strategy

$\forall i \in P, a_i \in A_i$ is player i's **pure strategy**

This strategy is referred to as pure, because there's no probability involved. For example, he always chooses *Scissors*.

Definition 2.1.4. Mixed strategy strategy is a probability measure of player's pure strategies. Set of player i's all mixed strategies is denoted as Σ_i

Mixed strategies allow a player to probabilistically choose actions. For example, his mixed strategy could be $(Rock - 0.4, Paper - 0.4, Scissors - 0.2)$

2.1.2 Normal form

If there are only two players ($|N| = 2$) and the game is finite , it can be conveniently described using the table (Fig.2.1). Rows correspond to actions of player one, columns to action of player two. In a cell (i, j), there are payoffs for both players - $(u_1(i, j), u_2(i, j))$

Examples

Follow normal form representations of the four previously mentioned games.

	Rock	Paper	Scissors
Rock	(0, 0)	(-1, 1)	(1, -1)
Paper	(1, -1)	(0, 0)	(-1, 1)
Scissors	(-1, 1)	(1, -1)	(0, 0)

(a) Rock-Paper-Scissors

	Rock	Paper	Scissors	Lizard	Spock
Rock	(0, 0)	(-1, 1)	(1, -1)	(1, -1)	(-1, 1)
Paper	(1, -1)	(0, 0)	(-1, 1)	(-1, 1)	(1, -1)
Scissors	(-1, 1)	(1, -1)	(0, 0)	(1, -1)	(-1, 1)
Lizard	(-1, 1)	(1, -1)	(-1, 1)	(0, 0)	(1, -1)
Spock	(1, -1)	(-1, 1)	(1, -1)	(-1, 1)	(0, 0)

(b) Rock-Paper-Scissors-Lizard-Spock

	Confess	Be quiet
Confess	(8, 8)	(0, 10)
Be quiet	(10, 0)	(2, 2)

(c) Prisoner's dilemma

	Opera	Football
Opera	(-8, 10)	(0, 0)
Football	(-20, -10)	(10, -6)

(d) Battle of the sexes

Figure 2.1: Few strategic games in normal form. Note that some games are zero sum games - (a) and (b); some are not - (c) and (d)

2.2 Extensive form games

In many games, players don't act simultaneously, but rather sequentially make some actions. This is the case of Chess, Tic-Tac-Toe, Poker and many more.

Extensive form games model this behavior by using a tree-like game representation. Nodes correspond to game states, edges to player's actions. In every node, the corresponding player chooses some action (one of the outgoing edges). Leafs correspond to final states and contain utilities for all players.

I already mentioned that in some games, players have *imperfect information*. Extensive form games also allow to model this property. Having a game with imperfect information, some game states are indistinguishable from player's point of view. Those states are grouped to so called *information sets*. Player's decisions are then made in these *informatin sets* rather then in game states.

Extensive form games also allow to model actions of "nature". Another player, so called chance player, is part of the game and chooses his actions according to some probability distribution. In Poker, the chance player randomly deals cards.

Definition 2.2.1. Finite extensive form game with imperfect information Is a tuple $\langle N, H, P, f_c, (I), u \rangle$, where [29]

- A finite set N (the set of players)
- A set H of sequences (possible histories), such that:

  empty sequence is in H

  every prefix of sequence in H is also in H

  $Z \subset H$ of terminal histories (those which are not a prefix of any other history)

  The set of action available in every non-terminal history h is denoted $A(h) = a : (h, a) \in H$
- A function P that assigns to each non-terminal history a member of $N \cup c$. (P is the player function, $P(h)$ being the player who takes an action after the history h. $If P(h) = c$ then chance determines the action taken after history h)
- A function f_c that associates with every history for which $P(h) = c$ a probability measure on $A(h)$, where each such probability measure is independent of every other such measure
- For each player $i \in N$ a partition $\mathcal{I}_i$ of $h \in H$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$. $\mathcal{I}_i$ is the **information partition** of player i ; a set $I_i \in \mathcal{I}_i$ is an **information set** of player i
- For each player $i \in N$ a **utility function** $u_i : Z \to R$. I refer to this function also as an **outcome function** O_i

2.2.1 Simple card game example

As a simple example, I will show *extensive form game* formulation of very simple card game. This can be seen as a very small Poker game.

The rules

1. Player one is dealt either Ace or King
2. Player two is dealt either Ace or King
3. Pot (money on the table players are playing for) is two dollars
4. Player one acts

   can either bet one dollar or check
5. Player two acts

   can bet (if player one checked), or if player one bet, he can either call (even up the bet) or fold (give up)
6. Player one acts again only if he checked first and second player bet

   can either call or fold
7. Player who folded or had worse card looses one dollar

Extensive form game formulation

1. There are two players - $N = \{1, 2\}$.
2. The set H of all histories
   $H = \{(K), (K, A), (K, A, bet), (K, A, bet, call), \ldots\}$
3. The set Z of terminal histories
   $Z = \{(K, A, bet, call), (K, A, bet, fold), (K, K, check, bet, fold), \ldots\}$
4. The player function
   $P(\emptyset) = c$, $P(K) = c$, $P(K, A) = 1$, $P(K, A, check) = 2$,
   $P(K, A, check, bet) = 1$ etc.
5. Information sets - we need to group histories that are indistinguishable by the player (he can't see opponent's card).
   $\mathcal{I}_1 = \{\{(K, A), (K, K)\}, \{(K, A, bet), (K, K, bet)\}\}, \ldots\}$,
   $\mathcal{I}_2 = \{\{(K, A), (A, A)\}, \{(K, A, bet), (A, A, bet)\}\}, \ldots\}$
6. The utility function on terminal histories.
   $u_1(K, A, check, bet, fold) = -2$, $u_2(K, A, check, bet, fold) = +2$,
   $u_1(K, A, check, bet, call) = -3$, $u_2(K, A, check, bet, call) = +3$

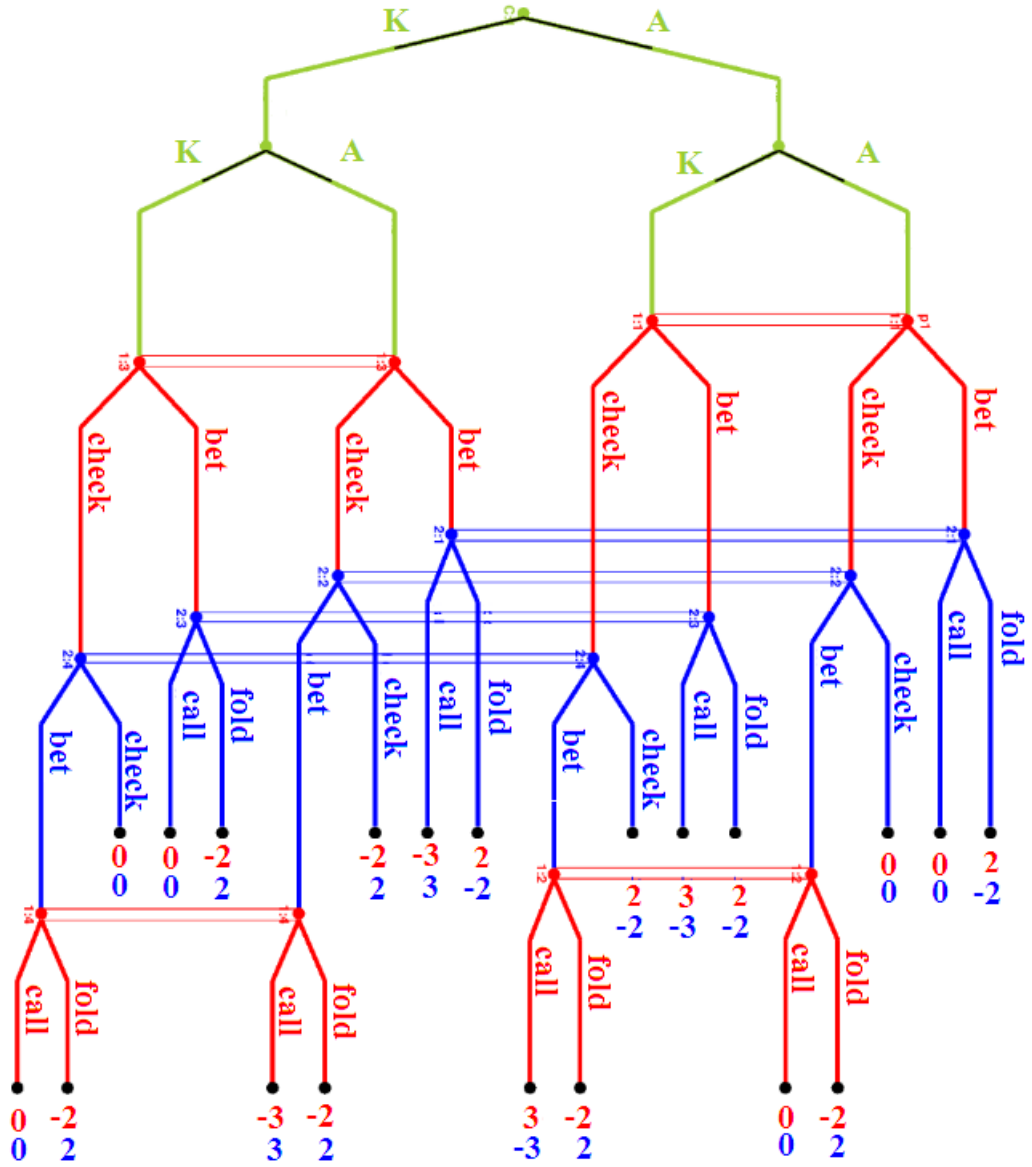


Figure 2.2: Nodes grouped in information set are connected using horizontal lines. Picture created using the Gambit open source software.

Let me briefly describe what is presented on 2.2. Green color corresponds to the chance player - in the first two tree levels, cards are dealt. In each node, there are some labeled action that player can take (red color for the first player, blue for the second one). Notice that some tree nodes are connected using horizontal line - this indicates what nodes are grouped in an information set. In this game, information sets allow us to describe the fact, that the players don't see what the opponent was dealt.

We group nodes the player can't distinguish (can't see opponent's card), to information sets, enforcing the player to play the same strategy in these nodes.

The utilities are visible in every leaf. For the path $(A - K - bet - call)$, red player's utility is 3, blue player's utility is -3 (red player was dealt better and won).

2.2.2 Perfect and Imperfect recall

Information sets can not only be used to model games where a player doesn't see some hidden information, but can also group nodes so that the player may "forget" some information he already knew. If this is the case, it's referred to as *games with imperfect recall* (and *games with perfect recall* otherwise).

Formally, let $\langle N, H, P, f_c, (I_i)\rangle$ be an extensive game form and let $X_i(h)$ be the record of player i's experience along the history h: the sequence consisting of the information sets that the player encounters in the history h and the actions that he takes at them. Extensive game form has **perfect recall** if for each player i we have $X_i(h) = X_i(h')$ whenever the histories h and h' are in the same information set of player i.

2.2.3 Strategies

Definition 2.2.2. A **mixed strategy** of player i in extensive form game G is a probability measure over the set of player's pure strategies.

Definition 2.2.3. A **behavior strategy** of player i is a collection $(\beta(I_i))$ of independent probability measures, where $\beta(I_i)$ is the probability measure over $A(I_i)$. For any history $h \in I_i \in H$, any action $a \in A(h)$ we denote by $\beta(h, a)$ the probability assigned by $\beta(I_i)$to the action a.

Intuitively, mixed strategies allow the player to mixed between all his possible paths (pure strategies). On the other hand, behavior strategy allows player to mix his strategies in every information set. There's a following relation between these two type of strategies:

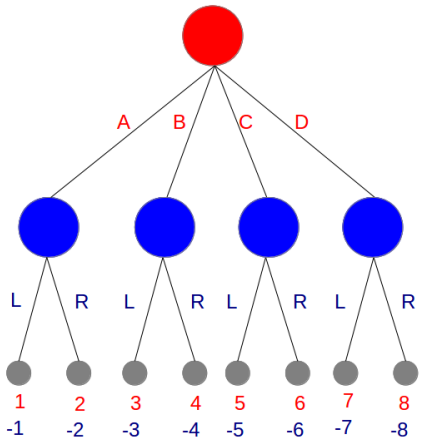
Theorem 1. *For extensive form games with perfect recall, mixed and behavior strategies are equivalent [29][p. 203].*

2.2.4 Normal form of extensive form game

Given any game in normal form, it's easy to create an equivalent extensive form game. The root node corresponds to player one's decision, and all next level nodes are grouped to the information set of player two (this way we simulate simultaneous turns - player two doesn't know what action player one chose). But is it possible to convert any two-player extensive from game to equivalent game in normal form? It's not possible in general, but:

Lemma 1. Given any two-player extensive form game with perfect recall, it's possible to create an equivalent normal form game [29].

We can create the normal form game by considering all possible information sets and actions - all possible pure strategies. Unfortunately, the created normal form game can be exponentially large compared to extensive form game representation.



(a)

	A	B	C	D
(A-L, B-L, C-L, D-L)	1;-1	3;-3	5;-5	7;-7
(A-L, B-L, C-L, D-R)	1;-1	3;-3	5;-5	8;-8
(A-L, B-L, C-R, D-L)	1;-1	3;-3	6;-6	7;-7
(A-L, B-L, C-R, D-R)	1;-1	3;-3	6;-6	8;-8
(A-L, B-R, C-L, D-L)	1;-1	4;-4	5;-5	7;-7
(A-L, B-R, C-L, D-R)	1;-1	4;-4	5;-5	8;-8
(A-L, B-R, C-R, D-L)	1;-1	4;-4	6;-6	7;-7
(A-L, B-R, C-R, D-R)	1;-1	4;-4	6;-6	8;-8
(A-R, B-L, C-L, D-L)	2;-2	3;-3	5;-5	7;-7
(A-R, B-L, C-L, D-R)	2;-2	3;-3	5;-5	8;-8
(A-R, B-L, C-R, D-L)	2;-2	3;-3	6;-6	7;-7
(A-R, B-L, C-R, D-R)	2;-2	3;-3	6;-6	8;-8
(A-R, B-R, C-L, D-L)	2;-2	4;-4	5;-5	7;-7
(A-R, B-R, C-L, D-R)	2;-2	4;-4	5;-5	8;-8
(A-R, B-R, C-R, D-L)	2;-2	4;-4	6;-6	7;-7
(A-R, B-R, C-R, D-R)	2;-2	4;-4	6;-6	8;-8

(b)

Figure 2.3: Extensive form game (a) and its corresponding normal form game representation (b).

2.2.5 The sequence form

Main idea behind sequence form representation is to represent all paths of the extensive form (instead of pure strategies, since there are exponentially many of them).

Definition 2.2.4. A **sequence** of actions of player $i \in N$, defined by history $h \in H$, is the ordered set of player i's actions contributing to that history. Let $\emptyset$ denote the sequence corresponding to the root node. The set of sequences of player i is denoted as Σ_i. $\Sigma = \Sigma_1 \times \ldots \times \Sigma_n$ is the set of all sequences. [11]

Definition 2.2.5. The **sequence payoff function** $g_i : \Sigma \to R$ for agent i is is given by $g(\sigma) = u_i(z)$ if a leaf node $z \in Z$ would be reached when each player played his sequence $\sigma_i \in \sigma$, and by $g(\sigma) = 0$ otherwise. [11]

Unlike the normal form, the size of the sequence form representation is linear in the size of the extensive form game. There's only one sequence for each game tree node (history). As we will see later, both normal and sequence forms can be used to compute optimal strategies using linear programming. But since the size of sequence form is linear (in contrast to exponential size of normal form), it's much more appealing.

	0	A	B	C	D
0	0;0	0;0	0;0	0;0	0;0
L_a	0;0	1;-1	0;0	0;0	0;0
R_a	0;0	2;-2	0;0	0;0	0;0
L_b	0;0	0;0	3;-3	0;0	0;0
R_b	0;0	0;0	4;-4	0;0	0;0
L_c	0;0	0;0	0;0	5;-5	0;0
R_c	0;0	0;0	0;0	6;-6	0;0
L_d	0;0	0;0	0;0	0;0	7;-7
R_d	0;0	0;0	0;0	0;0	8;-8

Figure 2.4: Sequence form of the game from 2.3a

2.3 Stochastic game

Intuitively speaking, stochastic game models situations where instead of one game(state), there is a collection of games and players repeatedly play games from this collection. Transition from a current game to another probabilistically depends on actions taken in the current game. Poker tournament is an example of *stochastic game*. Basically, individual games are single hands and correspond to different stacks. I will describe poker as a *stochastic game* later.

Definition 2.3.1. A stochastic game G is a tuple $\langle N, S, A, p, r\rangle$ where

- $N = \{1 \dots n\}$ denotes a finite set of players
- S denotes a set of states(games)
- A is a tuple $(A_1, .., A_n)$ where for $s \in S$, A_i(s) is a finite set of player i's actions at state s
- $p_{s,t}(a)$ denotes the probability of transformation from state s to t when all players play according to the strategy profile $a \in A(s)$
- $r : S \to \mathbb{R}^n$ denotes the payoff function, where i'th component is the payoff to the player i when the state s is reached

State s is *terminal* if $p_{s,s'}(a) = 0$ for all $s' \neq s, a \in A(s)$ (the game is over). A state is *nonterminal* if it is not *terminal*. Terminal states in poker tournament are states when players are eliminated (and winners receive payouts according to the payout structure)

2.3.1 Strategies

Mixed strategy of player $i \in N$ in a state $s \in S$ is a probability distribution over actions in $A_i(s)$, denoted as $\sigma_{i,s}$. For the current state s_t, the goal of player i is to maximize $\sum_{t=0}^{\infty} \gamma^t r_i(s_t)$.

3. Nash equilibrium

I already described what Nash equilibrium basically is. Having defined some models, it's now possible to define this concept formally.

3.1 Finite strategic game

A Nash equilibrium of a strategic game $\langle N, (A_i), (u_i) \rangle$ is a profile $\sigma^* \in \Sigma$ with the property that:

$$u_i(\sigma^*_{-i}, \sigma^*_i) \geq u_i(\sigma^*_{-i}, \sigma_i) \; \forall \, i \in N, \sigma_i \in \Sigma_i \tag{3.1}$$

Which states that "no player can improve if the other players stick to their strategies". So given any strategy profile for any finite game, one can easily check if it's an equilibrium using the definition. We just check if all pure strategies of player i give the same utility as σ^*_i

3.2 Extensive form games

A Nash equilibrium in mixed strategies [29][p. 203](recall that for games with perfect recall, it's the same as behavior strategies) is a profile $\sigma^* \in \Sigma$ with the property:

$$O_i(\sigma^*_{-i}, \sigma^*_i) \geq O_i(\sigma^*_{-i}, \sigma\_i) \; \forall \, i \in N, \sigma_i \in \Sigma_i \tag{3.2}$$

This is essentially the same as 3.4. Following concept of sequential equilibrium enforces to play optimally each "subgames" as well.

3.2.1 Sequential equilibrium

If player plays optimally, he plays the "best strategy" given other players strategies. In extensive form games, player acts many times during the game (in contrast to *strategic games*). Intuitively, he should play optimally in any of his information sets.

This is basically the idea behind sequential equilibrium. But since players don't actually know what state the game is in (they only know the information set), formally defining this idea is not that straightforward. In Poker terms, if a player doesn't know opponent's cards, how could he know his counterstrategy?

Informally, one first assigns to every information set what he thinks about the actual game state ("if opponent raised 200$, I think 80% times he holds aces and 20% queens") - this is called *assesment*. Now given this belief, he needs to play "best response". Finally, we want the belief to be computed from opponent's strategies using Bayes' rule.

Assesment

An assessment in an extensive game is a pair (σ, μ), where β is a profile of behavioral strategies and μ is a function that assigns to every information set a probability measure on the set of histories in the information set.

The interpretation of μ, which we refer to as a **belief system**, is that $\mu(I, h)$ is the probability that player $P(I)$ assigns to the history $h \in I$, conditional on I being reached. For example in poker, this the probability we assign to the cards opponents may hold given their action ("what is the probability opponent is holding Aces given he raised?").

Define $O_i(\sigma, \mu|I)$ of (σ, μ) as denote the outcome conditional on I being reached.

Sequential rationality

The assessment (σ, μ) is sequentially rational, if for every player $i \in N$, every information set $I_i \in I_i$ and every strategy $\sigma_i' \in \Sigma_i$:

$$O_i(\sigma, \mu|I) \geq O_i((\sigma_{-i}, \sigma_i'), \mu|I) \tag{3.3}$$

Intuitively, sequential rationality states that each player plays optimally based on his belief (best response given some belief).

Consistency

Informally, consistency means that belief system is derived from strategy profile using Bayes' rule (so it's *consistent* with the strategy). Unfortunately, since every strategy must be defined in all information sets and there may be some information sets that are not reachable given a strategy, defining consistency is not straight-forward application of Bayes' rule

Let $\Gamma = \langle N, H, P, Fc, (I_i), (u_i) \rangle$ be a finite extensive form game with perfect recall. An assesment (β, μ) is **consistent** if there is a sequence $((\beta^n, \mu^n))_n^\infty = 1$ of assesments that converges to (β, μ) in Euclidian space and has the properties that each strategy profile β^n is completely mixed and that each belief system μ^n is derived from β^n using Bayes' rule. [29]

Sequential equilibrium

An assessment is a **sequential equilibrium** of a finite extensive game with perfect recall if it's *sequentially rational* and *consistent*. Every finite extensive game with perfect recall has a sequential equilibrium [29][p. 225]

As we will see later, sequential equilibrium allows to prove an interesting property regarding betting actions in no-limit Poker.

3.3 Epsilon-equilibrium

In Nash equilibrium, the player can't improve by changing his strategy. An ϵ-equilibrium states that he can improve only by some small ϵ

An epsilon-equilibrium is a profile $\sigma^* \in \Sigma$ with the property that:

$$u_i(\sigma^*_{-i}, \sigma^*_i) \geq u_i(\sigma^*_{-i}, \sigma_i) - \epsilon \; \forall \, i \in N, \sigma_i \in \Sigma_i \tag{3.4}$$

3.4 Best response

Knowing the strategy of other players, we know what are the "best counteractions". Those are the actions with the biggest utility against opponent's strategies. This is called **best response**.

For any $a_{-i} \in A_{-i}$ define $B_i(\sigma_{-i})$ to be the set of player i's pure best action given a_i. We refer to $b_i \in B_i(\sigma_{-i})$ as a **pure best response**

$$B_i(\sigma_{-i}) = \{a_i \in A_i : u_i(a_i, \sigma_{-i}) \geq u_i(a'_i, \sigma_{-i}) \; \forall a'_i \in A_i\} \tag{3.5}$$

Similarly, best response

$$r_i(\sigma_{-i}) = \{\sigma_i \in \Sigma_i : u_i(\sigma_{-i}, \sigma_i) \geq u_i(\sigma_{-i}, \sigma'_i) \, \text{for all} \, \sigma'_i \in \Sigma_i\} \tag{3.6}$$

Defining equilibrium via best responses

Reformulating "no player can't improve" to "all players play best response" gives alternative formulation of Nash equilibrium. A Nash equilibrium is a profile a^* of actions for which

$$a^*_i \in r_i(a^*_{-i}) \; \forall i \tag{3.7}$$

This alternative definition is the key element for the proof of existence of Nash equilibrium.

3.5 Existence of Nash equilibrium

3.5.1 Pure strategies

I haven't mentioned if this optimal strategy profile exists at all. If the players are restricted to pure strategies, we've actually already seen a game where this pure equilibrium doesn't exist.

Recall the strategic game rock-paper-scissors. Since there are only 3×3 possible pure Nash equilibriums (that is, 9 possible tuples of pure strategies), we can enumerate all of them and see that indeed none of them satisfies a definition of Nash equilibrium 3.4.

On the other hand, we have already seen a game where pure Nash equilibrium exists. It's easy to check that if both players choose to confess, the strategy profile forms the Nash equilibrium.

To sum it up, pure Nash equilibrium may or may not exist.

	Rock	Paper	Scissors
Rock	(0, 0)	(-1, 1)	(1, -1)
Paper	(1, -1)	(0, 0)	(-1, 1)
Scissors	(-1, 1)	(1, -1)	(0, 0)

Figure 3.1: Rock-Paper-Scissors, none of the cells form a Nash equilibrium

	Confess	Be quiet
Confess	(8, 8)	(0, 10)
Be quiet	(10, 0)	(2, 2)

Figure 3.2: Prisoner's dilemma - red cell forms a pure Nash equilibrium. We see that if any player changes his strategy (and the other one doesn't), his utility doesn't improve (here, it actually gets worse)

3.5.2 Mixed strategies

If all players are allowed to mix their strategies, Nash equilibrium is guaranteed to exist (for finite games). Define a set-valued function $r = (r_i(\sigma_{-i}), r_{-i}(\sigma_i))$. Then from 3.7 follows:

Lemma 2. a^* is a fixpoint of function $r \iff a^*$ is Nash equilibrium

This function satisfies all properties for the Kakutani fixed point theorem [3]. This gives us the key result of existence of Nash equilibrium for mixed strategies. Note that this is just proof of existence. Chapter 4 deals with computing these strategies.

Rock-Paper-Scissors As we have already seen, this game has no pure equilibrium. Now we know that mixed equilibrium - how does it look like? It turns out that it's $(Rock - \frac{1}{3}, Rock - \frac{1}{3}, Rock - \frac{1}{3})$ for both players. So player should randomly (uniform distribution) select his action. Again, we can easily check that this is indeed Nash equilibrium.

3.6 Some properties

3.6.1 Two players, zero sum

Nash equilibrium has one appealing property in this case. If (σ_1, σ_2) forms a Nash equilibrium, player has guaranteed utility if he plays his strategy. This utility is referred to as the **game value**. This property follows from the definition - since opponent can't improve, and players are strictly competitive, player always gains this value if he plays Nash.

3.6.2 Other cases

Above argumentation doesn't hold here. Even though opponent can't improve when all players play according to a Nash, utilities of other opponent's can arbitrary change. This even if players want to play Nash. Suppose that both $(\sigma_1, \sigma_2, \sigma_3)$ and $(\sigma'_1, \sigma'_2, \sigma'_3)$ form a Nash equilibrium. If players play $(\sigma_1, \sigma_2, \sigma'_3)$ (that is, all players play a strategy in Nash, but they didn't communicate which Nash should they choose), utilities gained by players can be arbitrary high or low. It may also be a case that there are many equilibrium profiles, and one player decides which one to play (strategies for other player are same in these equlibriums)

Such cases occur in real-life situations, as suggested by [39]. [39] found a family of equilibrium profiles for three-player Kuhn Poker. The profiles exhibit an interesting property where one player can shift utility between the other players, while staying in equilibrium and not changing his utility.

4. Computability

If there's a Nash equilibrium, how to compute one? Are there any efficient algorithms? The result of existence of Nash equilibrium in any finite strategic game is from 1951 [27], but the first results on computational complexity of Nash equilibrium are from late nineties. The result for complexity class of computing a Nash equilibrium is from 2006 [8].

I find it very exciting that many results mentioned in this thesis are so new. Some of those results also find straightforward application - that is, help us compute much larger games than ever before.

In this chapter, I present current results and some algorithms for computing Nash equilibrium.

4.1 Two players, zero sum

It turns out that this is the easiest case. The key observation is that both players are competitive - that is, one player wants the opponent to loose as much as possible (because this the amount he wins).

This is leveraged in a linear program that consequently leads to the optimal strategy. Unfortunately, if there are more players or the game is not zero sum, this property is lost and the problem seems to be much harder.

4.1.1 Linear program

Normal form games

For any finite, two player zero sum games in normal form, it's possible to create a linear program LP with these properties

1. s^* is an optimal solution to $LP \iff s^*$ is a Nash equilibrium
2. size of the LP is linear in the size of normal form

Extensive form games

As mentioned previously, extensive form games can be converted to the normal form. We could compute the equilibrium by using the normal form game, but since the size of normal form game can be exponential, we don't want to do that.

Fortunately, we have already seen the sequence form 2.4. This form can be used as well

1. s^* is an optimal solution to $LP \iff s^*$ is a Nash equilibrium
2. size of the LP is linear in the size of sequence form

Combining these results, we get the following corollary

Theorem 2. *There's an polynomial-time algorithm for finding a Nash equilibrium in two player, zero sum game (in both, extensive and normal form representation)*

4.1.2 Fictitious play

Fictitious play is an iterative algorithm, where players select their strategies based on previous iterations. At each iteration, players select the *pure best response* to the opponent's empirical mixed strategy (based on previous iterations).

Theorem 3. *If finite two players zero sum game is played repeatedly using fictitious play, empirical game value converges to the game value [32]*

As for the convergence of strategies, there is a following results.

Theorem 4. *If finite two players zero sum game is played repeatedly using fictitious play, empirical mixed strategies converges to the game value [32]*

There are actually other classes of games where the fictitious play is guaranteed to converge. Unfortunately, it doesn't converge even for some simple two players non zero sum games. Shapley provides some small games where fictitious play fails to converge [34].

Fictitious play is easy to implement - we only need to compute best response at each iteration. Unfortunately, we get negative results regarding the speed of convergence

Theorem 5. *In symmetric two-player constant-sum games, FP may require exponentially many rounds (in the size of the representation of the game) before an equilibrium action is eventually played. This holds even for games solvable via iterated strict dominance. [4]*

Despite this, fictitious play is very popular algorithm - easy to implement and typically converges fast to ϵ - equilibrium. Fictitious play is also used in games that are not two players zero sum. Even though not guaranteed to converge, we have following result:

Theorem 6. *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. [12]*

Unfortunately, if there are more players or the condition of zero sum is removed, the problem seems to be much harder to solve.

4.2 Other cases

If computing a Nash equilibrium in a case of two players zero sum is easy, what about three, four or five players? Is it "harder" to compute a game with four players than a game with only three players? Do the problems fall to different computability classes? Latest results show that this is not the case.

Theorem 7. *Finding a Nash equilibrium in an r-player can be reduced to finding a Nash equilibrium in a game with 4 players [18](2005)*

4.2.1 Appropriate complexity class

So is it possible that finding a Nash equilibrium in a game with 4 players is NP-complete? It turns out that NP-completeness is not appropriate class of complexity. Basically, the reason is that for NP-complete problem, the solution may or may not exist. On the other hand, Nash equilibrium is guaranteed to exist. But there are many closely related problems without the guarantee of existence and those problems indeed fall to the class of NP-complete problems - this result is from 1989 [14]

- Does the game have at least two Nash equilibrium?
- Does the game have a Nash equilibrium with player i's utility at least a given amount?
- And many more...

PPAD complexity class

PPAD stands for "polynomial parity argument (directed case)" [30]. I won't formally define the class itself, but merely mention some of the interesting properties. It's a class of combinatorial problems, where the existence of solution is guaranteed. Solution corresponds to finding some specific vertex in an exponentially large graph with some specific properties.

- The graph is directed
- Every vertex has at most one incoming and outgoing edge
- It's easy to find neighbors (with the edge direction)
- We need to find any vertex with no outgoing edges (sink) - this is the solution.

There is no known polynomial time algorithm for this complexity class. Note that the existence of polynomial time algorithm doesn't imply $P = NP$. It could be that $PPAD = P$ and still $P \neq NP$

Finally, the problem of computing Nash equilibrium falls to this class. First, the case for 4 and more players was proven, shortly after followed by the rest.

Theorem 8. *The problem of computing Nash equilibrium in games with 4 players is PPAD-complete. [8](2006)*

Theorem 9. *The problem of computing Nash equilibrium in a bimatrix games is PPAD-complete. [6](2006)*

4.2.2 Summary

Computing a two players, zero sum game can be done using linear programming in polynomial time. As we will see, this doesn't mean that computing poker game with two players is tractable (Poker is simply too large). On the other hand, for small games with three or more players, we are usually able to compute reasonably good strategies (player can't win much more by deviating from his strategy). This is due to the fact that some algorithms which are guaranteed to find a Nash equilibrium in two players games, work usually well in other cases and often converge to good strategies.

5. Poker

Online Poker gained a huge popularity in last few years - revenues grew from $82.7 million in 2001 to $2.4 billion in 2005 [41]. There are many Poker cardrooms worldwide, with Pokerstars being the largest and most popular one (having almost 47 million registered users [31]).

Looking at these numbers, it's not surprising that many people got interested in creating some Poker programs that could beat their opponents.

On the other hand, Poker is an interesting game from the theoretical point of view. The game of poker has been identified as a beneficial domain for current AI research [2] In the last few years, new great results and algorithms were discovered. I would like mention Computer Poker Research Group (CPRG) of University of Alberta and the group from Carnegie Mellon University (CMU). Their results from the last 14years allowed to create most sophisticated and best performing Poker bots ever, and these results apply to all extensive form games.

5.1 Rules

Poker is a family of card games - there are many different variants and variations. In general, all players receive some *private cards*. Then there are some betting rounds, where players bet their money (or chips) according to some rules.

In those betting rounds, some public cards may be revealed, in other variants player may change their private cards. Player may "give up" (fold) during the game, but if there are some players who make it to the end, they reveal their public cards (showdown).

Based on private and public cards, winner is determined using some ordering over cards.

There are many different groups and variants of Poker. In this thesis, I deal only with the most popular one [40]. - *Texas Hold'em Poker*

Texas Hold'em Poker Each player is dealt two private cards (*hole cards*). There are four betting rounds - *Preflop*, *Flop*, *Turn* and *River*. Five public cards are subsequently dealt - three public cards on *Flop*, one card on Turn and one card on River.

First, mandatory bets are posted, known as *blinds* (*big blind* and *small blind*). In each betting round, players can *fold* (give up the pot - game ends), *check/call* (match the opponent's bet), *bet/raise* (bet more money). Betting round ends when both players' bets are equal and both of them made some action. If none of the players folds during all four rounds, one player wins (takes the pot) or it's a draw (pot is splitted). The winner is the player with the strongest 5-card hand. That is, he selects five cards from his two private hands and five public cards (seven in total) with the highest value. Note that these values (*poker hands*) form an ordering.

Please refer to [36] for more detailed rules. Follows some poker terminology used in the following chapters

- **preflop, flop, turn, river:** betting rounds
- **heads up (HU)**: there are two players
- **stack**: player's money/chips
- **pot**: money/chips on table
- **all-in**: bet equal to the size of player's stack (he bets all the money he has)
- **hand**: player's cards
- **limit betting/limit poker**: player is allowed to bet only a fixed amount
- **no-limit betting/no-limit poker**: player is allowed to bet any amount up to his stack

Figure 5.1: Heads-up no-limit Hold'em Poker, 1000$ table on PokerStars - that is, the winner wins 1000 Player on the left won the round, since his best 5-card hand is stronger than the opponent's
(K♢ A♡ A♠ 8♢ 2♢) $\succeq$ (9♡ J♡ A♠ 8♢ 2♢)

5.2 Poker bots

5.2.1 Computer vs humans

In contrast to chess, where state-of-the-art chess agents beat the world best chess players [10], poker agents have still long way to go (namely the no-limit variants).

In 2005, chess computer *Hydra* beat British grand master Michael Adams $(5-1-0)$ (won-draw-lost). In 2006, *Deep Fritz 10* beat Kramnik, World Champion at the time, $(4-2-0)$.

Compare it to no-limit Hold'em Poker. In the First Man-Machine Poker Championship (2007), the match between the program *Polaris* and two professional Poker players ended up $(1-1-2)$, humans won. In the Second Man-Machine Poker Championship (2008), advanced version of *Polaris* challenged four professional Poker players. *Polaris* won $(2-1-1)$

It's important to mention that there was a great progress in computing poker strategies since 2008. I believe that if the Man-Machine Poker Championship was to be held again, humans would stand no chance.

5.3 Computer vs computer

Computer poker competition is held every year since 2006. In 2013, there were 13 different agents in the heads-up limit Texas hold'em, 11 agents in the heads-up no-limit and 5 agents in the 3-player limit competition. Agents were submitted by a mixture of universities and individual hobbyists from 10 different countries around the world. [7]

We send our own agent for this year - see chapter 8 for details.

5.3.1 Comparing opponents

Note that in chess, we can play only few games between human and computer to determine who won. To get some statistical significance, one needs to play much more matches.

This is not the case with Poker. Even though there are techniques to decrease the necessary number of matches [23], one still needs to play thousands of hands to get some statistical significance.

This seems almost impossible in real life tournaments. On the other hand, it's easy to get these numbers playing online. This is thanks to two fact. First, Poker players typically play many tables simultaneously - client software allows having many table windows and brings the table to the front whenever player is to act. Second, every hand is much faster - dealing cards is faster as well as player's actions. Consequently, professional players are able to play enough hands in a year to easily overcome the variance.

6. Computing Poker strategies - Part I

In this chapter, we will finally see some application of game theory to Hold'em Poker. I will show some *minigames* that are easy to solve. Some of these minigames occur in real life games, typically in the end of tournaments.

First notice that Poker is always a *zero sum* game. This follows from the fact that all the money (or chips) player wins, other players looses.

6.1 Heads-up push fold

Consider a poker game with following properties:

1. Two players (heads up)
2. No limit betting
3. Both players have stack lower than some small multiplier of big blinds - K

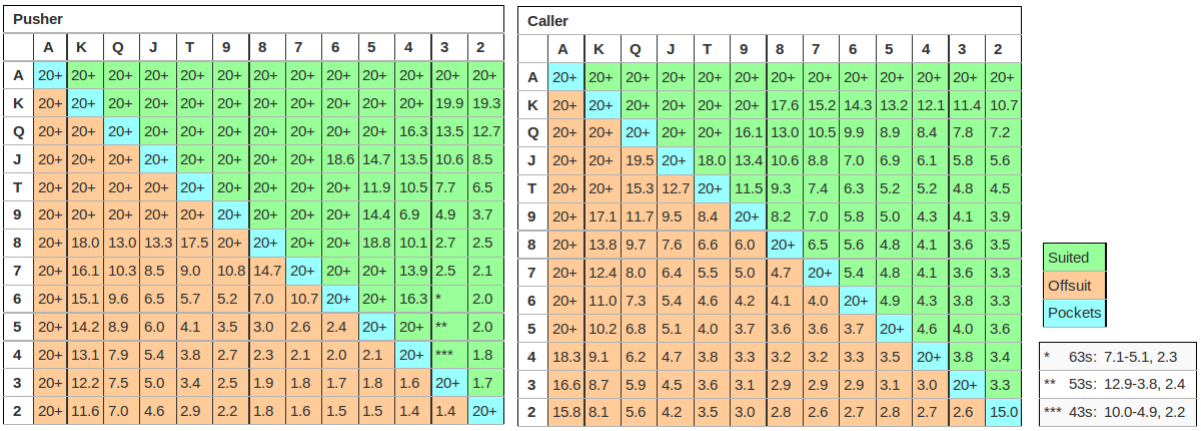
Now having two players zero sum game, it's theoretically possible to compute this game in polynomial time (2). Unfortunately, the game is too large, making the computation intractable.

But if the K is very low, it's possible to play almost optimally using only three actions - *all-in*, *fold*, *call* [5]. This makes the game tree much smaller, because there are no more actions possible - one of the players folds or one of the players is all-in. When a player is all-in, public cards are dealt and player with better combination wins the pot (or they draw).

How does the game tree look like? First, both players are dealt their private cards and pushing/folding follows. There two types of leafs: either one of the players folds (and looses his bet), or one of them is all-in. The utilities equal to expected value of chips (based on both player's hands). This game is so small that it's possible to represent it using the sequence form and compute using linear programming.

I used both GLPK and Gurobi and the computation took only fraction of second using my computer. There is also a nice web interface from Holdem-Resources, which uses described method - see citeholdemresources. This way, players can easily see the optimal strategy for the push-fold heads-up minigame.

Even more, it's possible to represent the strategy in a simple table. Players can see in this table if they should *push* or *fol*d based on his hand and K.



Pusher

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+
K	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	19.9	19.3
Q	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	16.3	13.5	12.7
J	20+	20+	20+	20+	20+	20+	20+	20+	18.6	14.7	13.5	10.6	8.5
T	20+	20+	20+	20+	20+	20+	20+	20+	20+	11.9	10.5	7.7	6.5
9	20+	20+	20+	20+	20+	20+	20+	20+	20+	14.4	6.9	4.9	3.7
8	20+	18.0	13.0	13.3	17.5	20+	20+	20+	20+	18.8	10.1	2.7	2.5
7	20+	16.1	10.3	8.5	9.0	10.8	14.7	20+	20+	20+	13.9	2.5	2.1
6	20+	15.1	9.6	6.5	5.7	5.2	7.0	10.7	20+	20+	16.3	*	2.0
5	20+	14.2	8.9	6.0	4.1	3.5	3.0	2.6	2.4	20+	20+	**	2.0
4	20+	13.1	7.9	5.4	3.8	2.7	2.3	2.1	2.0	2.1	20+	***	1.8
3	20+	12.2	7.5	5.0	3.4	2.5	1.9	1.8	1.7	1.8	1.6	20+	1.7
2	20+	11.6	7.0	4.6	2.9	2.2	1.8	1.6	1.5	1.5	1.4	1.4	20+

Caller

	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+	20+
K	20+	20+	20+	20+	20+	20+	17.6	15.2	14.3	13.2	12.1	11.4	10.7
Q	20+	20+	20+	20+	20+	16.1	13.0	10.5	9.9	8.9	8.4	7.8	7.2
J	20+	20+	19.5	20+	18.0	13.4	10.6	8.8	7.0	6.9	6.1	5.8	5.6
T	20+	20+	15.3	12.7	20+	11.5	9.3	7.4	6.3	5.2	5.2	4.8	4.5
9	20+	17.1	11.7	9.5	8.4	20+	8.2	7.0	5.8	5.0	4.3	4.1	3.9
8	20+	13.8	9.7	7.6	6.6	6.0	20+	6.5	5.6	4.8	4.1	3.6	3.5
7	20+	12.4	8.0	6.4	5.5	5.0	4.7	20+	5.4	4.8	4.1	3.6	3.3
6	20+	11.0	7.3	5.4	4.6	4.2	4.1	4.0	20+	4.9	4.3	3.8	3.3
5	20+	10.2	6.8	5.1	4.0	3.7	3.6	3.6	3.7	20+	4.6	4.0	3.6
4	18.3	9.1	6.2	4.7	3.8	3.3	3.2	3.2	3.3	3.5	20+	3.8	3.4
3	16.6	8.7	5.9	4.5	3.6	3.1	2.9	2.9	2.9	3.1	3.0	20+	3.3
2	15.8	8.1	5.6	4.2	3.5	3.0	2.8	2.6	2.7	2.8	2.7	2.6	15.0

Figure 6.1: Table cell denotes a number of blinds player should push with. If player's stack is lower or equal this number, player should go all-in with the corresponding hand. For example in the left table, we see that that the first player should go all-in with 7♠9♡ offsuit if his stack is ≤ 10.8. Source: [19]

6.2 Tournament push fold

If there are only two players, player just maximizes his expected stack after the hand. This is because the more chips/money the player have, the better. This may not be the case in tournaments. Consider this tournament situation:

- There are three players left in the game
- Payout structure is following

  First position (the winner) gets 50%

  Second position gets 50%
- The players are named A, B, C

And consider two different situations (possible chips distributions):

1. $(100, 100, 100)$
2. $(10, 1, 289)$

In the first distribution, player A has 100 blinds compared to 10 blinds in the second distribution. But because it doesn't matter if he finishes first or second (see the payout structure), the second distribution is actually better. This is because in the second distribution, players B have only 1 blind and is very likely to loose the game soon.

6.2.1 Computing tournaments

Now it's not possible to represent the game as in previous case, where leafs corresponded to expected stack. We can't maximize our value just in on hand (because we don't know which chip distribution is better). Nodes representing the end of hand can't be leafs (as were in the HU), but the game continues with another hand (blind position is changed and cards are dealt again). This way we get a stochastic game.

This was of course also the case with two players, but we could forget the next hand and just maximize stack the player would have for the next round. I present a picture for better illustration (6.2)

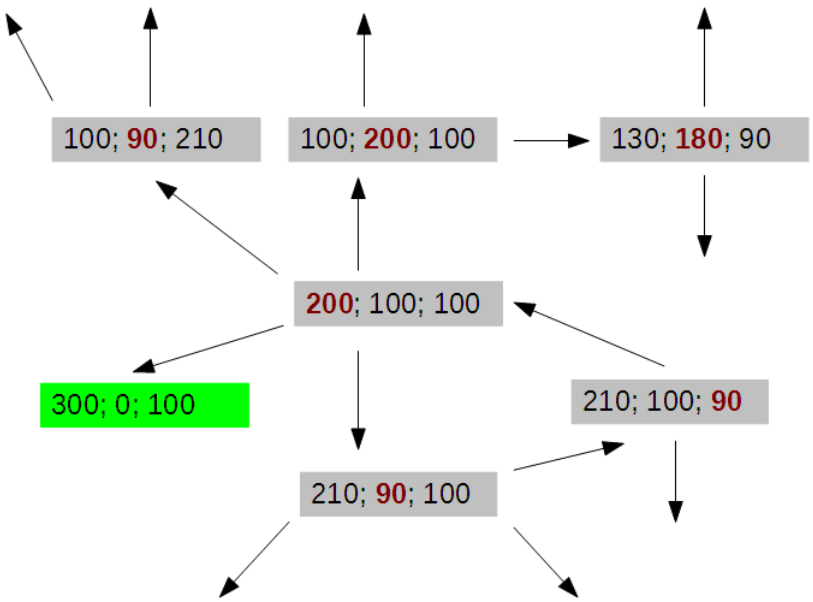


Figure 6.2: Three players tournament. Triples correspond to players' stack sizes. Red stack represents a player posting a big blind. Only few nodes and edges are presented. The green node is a final state - one of the players lost all chips and the game is over (so there are no outgoing edges from this state)

6.3 Independent chip model

Independent chip model (ICM) is a formula to estimate a player's game value based on the chips distribution and payout structure. If the game state values are estimated, it's possible to solve the game independently in any game state (as it was possible in heads up). It's clear that if the estimate is wrong, computed game strategy could be wrong as well. As I will show later, ICM is very good estimate in most of the situation.

ICM assumes that the probability of player i winning the game (finishing first) equals to $\frac{s_i}{\sum_{i'} s_{i'}}$ (s_i being player i's stack). Probability that player finishes on the second position equals to probability that any other player finished first and he finishes first among the remaining players (and so on for n-th position).

Stacks	**Expected values**
$(100, 100, 100)$	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$
$(10, 1, 289)$	$(0.37, 0.04, 0.59)$
$(1, 1, 298)$	$(0.2, 0.2, 0.6)$

Table 6.1: Few values computed via ICM. Stacks are displayed relatively to blinds, payout structure: $(0.6, 0.4)$

Thus, using this formula, one can compute player's expected value over all payout positions. Computing this formula from definition has exponential time complexity, but there's a simple dynamic algorithm that is polynomial.

Unfortunately, this formula doesn't handle blind position. It computes the same value for all players in this situation: $(1, 1, 1)$. Clearly, if all players have only one blind, their game value depends a lot on whether they do or don't post a blind.

6.3.1 Computing an equilibrium

Even with the state values estimated, it's hard to compute an equilibrium in a game with three players (see the computability chapter).

As mentioned in chapter 4, fictitious play can be used in these games and typically converges to reasonable good strategies. Fictitious play needs to compute best response in every iteration. This may seem hard, since there are 169 cards and player can push or fold with any one of them, making 2^{169} pure strategies.

But to find a best response, it's not necessary to try all pure strategies. The best response can be computed by simple traversing an extensive game tree.

Using the ICM and the fictitious play, we can compute strategies in tournaments with more than two players (we still limit players to push/fold strategies). There is a tool by HoldemResources [19] that computes these strategies. When we implemented this approach, we got the same results.

6.4 Approximate push fold equilibrium without ICM

[13] showed how to compute an approximate equilibrium in a tournament game. Main idea is to compute state values from a strategies in every state, recompute the strategies using those values and repeat. The algorithm works basically as follows:

1. Estimate state values using ICM
2. Run fictitious play in every state
3. Recompute state values using new strategies
4. Compute how much the values changed in step 3
5. If the change is bigger than some δ, go to step 2

This algorithm is not guaranteed to converge. Unfortunately, even if it converges, it may not converge to a Nash equilibrium. In the follow-up paper, they came up with an "ex post" check, which computes best response. This allows to check if the strategy profile is indeed an ϵ-equilibrium. In that paper, they also showed how to adapt the algorithm that if it converges, it converges to an equilibrium.

In their setup, there were 13500 total chips, small blind was 300 and big blind 600. The payout structure structure was: first place: 50%, second place: 30% and third place: 20%. Using described algorithm and "ex post" check, they computed an ϵ-equilibrium where none of the players was able to improve his value in state by more than 0.5% of the tournament entry fee.

The results also show that ICM is a very good estimate in most of the situations.

When I implemented this algorithm, it converged in this setup as well. Using different payout structure - 60/40/0, I was not able to force the algorithm to sufficiently converge

7. Regret algorithms

Learning algorithms make repeatedly decisions and adapt the current strategy based on previous results. This can be seen as repeatedly playing some game from one player's point of view. For the Rock-Paper-Scissors, player would select his action probabilisticaly on a previous results. For some traveling game, player would prefer paths that were previously fast. Preferring actions that previously performed better is the basic idea behind all these algorithms.

First, consider repeatedly playing a sequential, two player, zero-sum game G. Let σ_i^t be the strategy of player i on round t.

Definition 7.0.1. Average utility
Is just player's averaged utility over all games

$$\overline{u}_i^T = \frac{1}{T}\sum_{t=1}^{T} u_i(\sigma^t) \tag{7.1}$$

Definition 7.0.2. Average overall regret
(Also known as *external regret*) of player i at time T is the average value player would gain by playing single strategy σ_i^* all the time, that would maximize his average value (note that this value may be negative).

$$R_i^T = \frac{1}{T} max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t)) \tag{7.2}$$

Definition 7.0.3. Average strategy

$$\overline{\sigma}_i(a) = \frac{\sum_{t=1}^{T} \sigma_i^t(a)}{T} \tag{7.3}$$

In a case of two players, zero-sum game, there's a direct connection between the external regret and epsilon-equilibrium

Theorem 10. *If both player's external regret at time T is less than ϵ, then $\overline{\sigma}^T$ is a 2ϵ equilibrium*

Proof. Since the game is zero sum, $u_1 = -u_2$ and suppose that the player zero is maximizing.. Suppose for contradiction that there is a strategy σ_0^*, so that $u_0(\sigma_0^*, \overline{\sigma}_1) - u_0(\overline{\sigma}) > 2\epsilon$. Because the player one's external regret is less than ϵ, $\frac{1}{T}\sum_{t=1}^{T}(u_0(\overline{\sigma}_0, \sigma_1^t) - u_0(\sigma^t)) > -\epsilon$ and consequently $\frac{1}{T}\sum_{t=1}^{T}(u_0(\sigma_0^*, \sigma_1^t) - u_0(\sigma^t)) > \epsilon$ which is contradiction.

□

Consequently, to find an ϵ-equilibrium, it's possible to use some algorithm that minimizes average overall regret.

7.1 Regret minimization

There are many regret minimizing algorithms. To find an ϵ-equilibrium, regret grow must be bounded by some function so the average overall regret goes to zero. I will show a simple algorithm and using this algorithm, grow of external regret is bounded by $\sqrt{T}$. Consequently, average overall regret is bounded by $\frac{\sqrt{T}}{T} = \frac{1}{\sqrt{T}}$

7.1.1 Regret matching

Definition 7.1.1. Action regret

$$R_i^T(a) = \frac{1}{T} \sum_{t=1} T(u_i(a, \sigma_{-i}^t) - u_i(\sigma)) \tag{7.4}$$

Definition 7.1.2. Action positive regret

$$R_i^{T,+}(a) = max(R_i^T(a), 0) \tag{7.5}$$

Regret matching uses following equation to select a strategy a time $t + 1$

$$\sigma_i^{T+1}(a) = \frac{R_i^{T,+}(a)}{\sum_{a' \in A} R_i^{T,+}(a')} \tag{7.6}$$

if $\sum_{a' \in A} R_i^{T,+}(a') > 0$ and uniformly otherwise.

Theorem 11. *If $|u| = max_t \, max_{a,a' \in A} \, (u^t(a) - u^t(a'))$, and strategy at $t + 1$ is selected using (7.6), the average overall regret is bounded by*

$$R_i^T \leq \frac{|u|\sqrt{|A|}}{\sqrt{T}} \tag{7.7}$$

Note that this algorithm is easy to implement. All we need is to acummulate regrets at each iteration, and to compute simple fraction 7.6.

7.2 Counterfactual regret minimization

Since regret is defined using single strategy, we can use regret minimization algorithms on normal form games. It's also possible to use regret minimization algorithms directly on the extensive form game representation (in every information set). This result allows us to compute much larger extensive form games, and the paper was published in 2007 [44]. Briefly, we

1. Define *counterfactural regret* in information set
2. Bounding overall regret by the sum of all *counterfactural regrets*

7.2.1 Counterfactual regret

Counterfactual utility is the expected utility given that the information set I is reached and all players play using strategy σ except that player i plays to reach I. Let $\pi^\sigma(h, h')$ denote the probability of going from history h to history h' given σ.

Let Z_I be the subset of all terminal histories with a prefix in I and for $z\ in Z_I$ let $z[I]$ be that prefix. Define counterfactual value as [25]:

$$v_i(\sigma, I) = \sum_{z \in Z_I} \pi^\sigma_{-i}(z[I])\pi^\sigma(z[I], z)u_i(z) \tag{7.8}$$

Let $\sigma|_{I \to a}$ be a strategy profile identical to σ except that player i always chooses action a in information set I. Now similarly to action regret (using utilities), define **counterfactual regret** (using counterfactual utilities) as

$$R^T_{i,imm}(I) = \frac{1}{T} max_a \sum_t^T (v_i(\sigma|_{I \to a}, I) - v_i(\sigma^t, I)) \tag{7.9}$$

7.2.2 Bounding regret by counterfactual regret

Finally, there's a key relation between average overall regret and immediate counterfactual regret [44]:

$$R^T_t \leq \sum_{I \in \mathcal{I}_i} R^T_{i,imm}(I) \tag{7.10}$$

Consequently, we can minimize counterfactual regret in each information set to minimize average overall regret. Using 7.6 to minimize counterfactual regret, from 7.7 7.10 follows that

$$R^T_i \leq \frac{|\mathcal{I}_i||u|\sqrt{|A|}}{\sqrt{T}} \tag{7.11}$$

This algorithm is refered to as CFR or vanilla CFR.

7.3 Monte Carlo sampling

To compute counterfactural values and update the strategy, we traverse the game tree only once. If the game tree is too large and doesn't fit into the memory, we can "sample" only a part of the tree. Another possible advantage is that computing the sampled part of the tree is faster and possibly, this lower cost of each iteration may lead to faster convergence.

The key idea behind **Monte Carlo CFR**.is to sample a part of the tree, while having the expected value of immediate counterfactual regrets unchanged. [25].

First, we restrict the terminal histories Z we sample on each iteration. Let $\mathcal{Q} = Q_1, \ldots, Q_t$ be a set of subests of Z, for which $\bigcup_{i=\{1\ldots r} Q_i = Z$. One if these subsets is called a *block*. On each iteration, only one block is sampled (this block defines a subtree we walk).

Let q_J be the probability of sampling block Q_j. Let $q(z) = \sum_{j|z\in Q_j} q_j$ (the probability of sampling terminal history z). The **sampled counterfactual value** when updating block j is:

$$\tilde{v}_i(\sigma, I|j) = \sum_{z\in Q_j \cap Z_I} \frac{1}{q(z)} \pi^{\sigma}_{-i}(z[I])\pi^{\sigma}(z[I], z)u_i(z) \tag{7.12}$$

Notice that if we choose $Q = Z$ and $q_1 = 1$, we get the vanilla CFR. It's easy to show that the expected value of sampled counterfactural value matches counterfactual value.

$$E_{j\sim q_j}[\tilde{v}_i(\sigma, I|j)] = v_i(\sigma, I) \tag{7.13}$$

Monte Carlo Countefactual Regret Algorithms work by sampling a block at iteration and computing the *sampled immediate counterfactual regret*

$$\tilde{r}(I, a) = \tilde{v}_i(\sigma|_{I\to a}, I) - \tilde{v}_i(\sigma^t, I) \tag{7.14}$$

A member of family of MCCFR minimizing algorithms is specified by the set $\mathcal{Q}$. The algorithm samples a $Q \in \mathcal{Q}$, computes sampled immediate counterfactual regret in corresponding information sets, and applies regret matching to minimize regrets. First, we show that this approach (under certain conditions) minimizes overall regret for any member of the MCCFR family.

Let $\vec{a}_i$ be a subsequence of history a containing only player i's actions in that history. Let $\vec{A}_i$ be the se of all such subsequences. Let $I_i(\vec{(a)}_i)$ be the set of all information sets sets where player i's action sequence up to that information set is $\vec{(a)}_i$.

Define $B_i = \{I_i(\vec{a}) : \vec{a} \in \vec{A}_i\}$. Define $M_i = \sum_{B \in B_i} \sqrt{|B|}$ Using M_i, we can tighten the bounds of 7.11

$$R_i^T \leq \frac{M_i |u| \sqrt{|A|}}{\sqrt{T}} \tag{7.15}$$

Note that this is bound is indeed tighter, since $M_i \leq |I_i|$ (this bound can be realised on some games)

Having defined M, we can state the key result of MCCFR [25]:

For any $p \in (0, 1]$, when using MCCFR such that for all $Q \in \mathcal{Q}$ and $B \in \mathcal{B}$

$$\sum_{I \in B} (\sum_{z \in Q \cap Z_i} \frac{\pi^\sigma(z[I], z) \pi^\sigma_{-i}(z[I])}{q(z)})^2 \leq \frac{1}{\delta^2}$$

where $q(z) \leq \delta > 0$, then with probability at least $(1 - p)$:

$$R_i^T \leq (1 + \frac{2}{\sqrt{p}}) \frac{1}{\delta} \frac{M_i |u| \sqrt{|A|}}{\sqrt{T}}$$

This result holds for any member of MCCFR family, and we present a member with specific bounds

7.3.1 External-Sampling MCCFR

Here we sample only the actions of the opponent and chance, such that $q(z) = \pi^sigma_{-i}(z)$. The algorithm simply does a traversal of the game tree, sampling action at each history h for which $P(h) \neq i$. For the external-sampling MCCFR, following property holds:

For any $p \in (0,]$, when using external-sampling MCCFR, with probability at leas $(1 - p)$

$$R_i^T \leq (1 + \frac{\sqrt{2}}{\sqrt{p}}) \frac{M_i |u| \sqrt{|A|}}{\sqrt{T}}$$

So we get same order of iterations as for the vannila CFR. For games where players make roughly the same number of decisions, the iteration cost of the external-sampling MCCFR is $\Omega(\sqrt{|H|})$, while $\Omega(|H|)$ for vanilla CFR. Both algorithms require the same order of iterations (see **??** and 7.15).

7.3.2 Outcome-Sampling MCCFR

Here the blocks of $\mathcal{Q}$ are single terminal histories ($\forall Q \in \mathcal{Q}, |Q| = 1$). On each iteration, one terminal history is sampled and information sets along that history are updated. For this member, we get the following bounds [25]:

For any $p \in (0,]$, when using outcome-sampling MCCFR where $\forall z \in Z$ either $\pi^\sigma_{-i}(z) = 0$ or $q(z) \geq \delta > 0$ at every timestep, with probability at leas $(1 - p)$

$$R_i^T \leq (1 + \frac{\sqrt{2}}{\sqrt{p}}) \frac{1}{\delta} \frac{M_i |u| \sqrt{|A|}}{\sqrt{T}}$$

7.3.3 Experimental comparison

In [25], performance of Vanilla CFR, Vanilla CFR with pruning , Outcome-Sampling MCCFR and External-Sampling MCCFR were compared on four different games. (CFR with pruning simply prunes a subtree whenever the opponent has no probability of reaching corresponding history). See (Fig. 7.1) for the results.

1. Goofspiel bidding card games.
2. One-Card Poker a generalization of Kuhn Poker
3. Princess and Monster a pursuit-evasion game on graph.
4. Latent Tic-Tac-Toe simmilar to Tic-Tac-Toe, but the moves are discoled after the opponent's move (and lost if invalid)

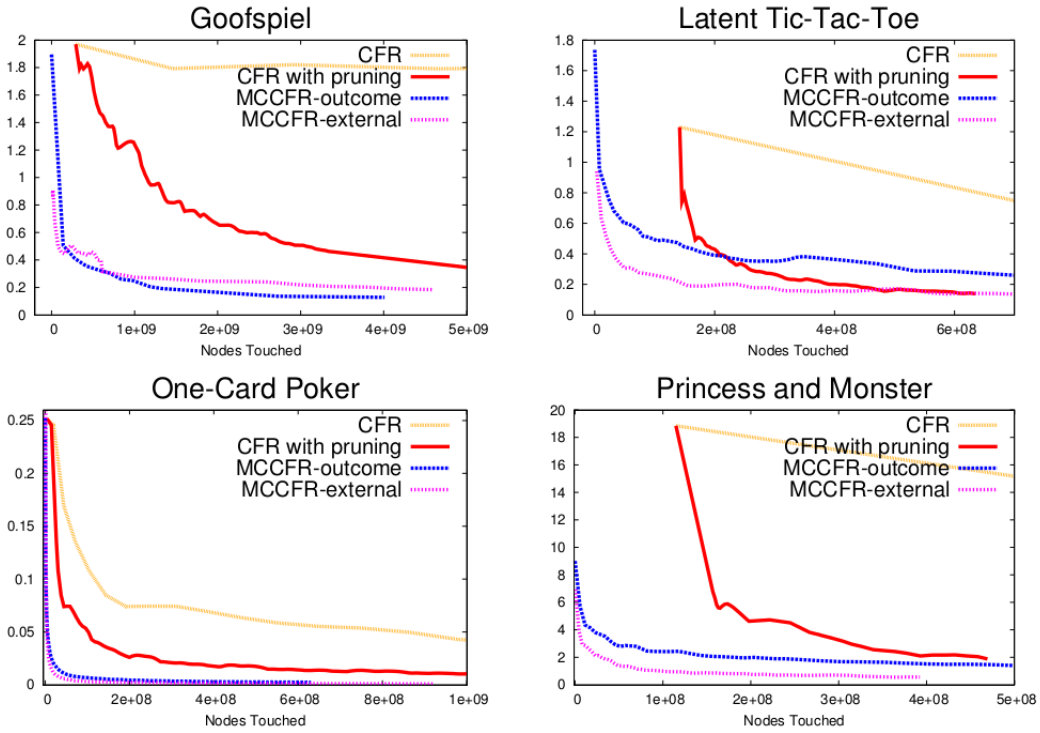


Figure 7.1: Experimental comparison of MCCFR members on four different games. Different colors correspond to different samplings. X-axis denotes the number of touched game tree nodes, Y-axis denotes the convergence (ϵ). We see that Vanilla CFR is outperformed in every game.

extraupUzavmmn extraup86 extraup87

8. Computing Poker strategies - Part II

All the poker game examples I gave used the fact, that the player could play push-fold strategy without loosing much value. This made the game small enough to compute. This chapter deals with poker games, where the push-fold argument holds no more (players have enough chips).

8.1 Size

Both limit and no-limit variants of poker can be represented using an extensive-form game representation. Size of this representation is limiting factor for computability purposes. In perfect information games, the game size could be represented simply by the number of game states. In the case of imperfect information, it's useful to know both - number of game states and information sets.

In limit variant of Poker, it's quite easy to compute those numbers, but it's not the case with no-limit variant. Size of this game depends on the players' stacks and blinds, and until 2013, was just estimated. For example, if both players have stack of 1000$ and the blinds are 1$/2$, the game was believed to have 10^{71} game states. Johanson [20] computed the size exactly for some no-limit cases. I created a table comparing a size of some popular games

Game	Game states	Inf. sets
Chess	10^{47}	-
Heads-up limit poker	3.162×10^{17}	3.194×10^{14}
Heads-no limit poker, $\$1000 - 1\$/2\$$	7.16×10^{75}	7.23×10^{72}
Heads-no limit poker, $\$1000 - 1\$/2\$$	7.16×10^{75}	7.23×10^{72}
Heads-no limit poker, $\$20000 - 50\$/100\$$	6.31×10^{164}	6.37×10^{161}

Because the number of game states for such a huge game was computed exactly, I can give the exact number of game states in a case of HU no-limit poker, $\$20000 - 50\$/100\$$. The number is: 631 143 875 439 997 536 762 421 500 982 349 491 523 134 755 009 560 867 161 754 754 138 543 071 866 492 234 040 692 467 854 187 671 526 019 435 023 155 654 264 055 463 548 134 458 792 123 919 483 147 215 176 128 484 600 [20]

8.2 Abstraction

Because the game size is far from tractable, it's necessary to apply some abstraction techniques to the game. The purpose of abstraction is to create smaller, tractable game, and to use the computed strategy for the original, unabstracted game.

To play the original game, the mapping between must be defined both ways:

- given any state in the original game, find a state in the abstracted game
- given player's action in the abstracted game, find an action in original game

In general, there are two different abstractions : *lossless* abstractions and *lossy* abstractions. Theoretically, the abstracted game could be created using any technique. Most straightforward technique to create smaller game from an extensive from game is to group some information sets to shrink the game. This is the case of all poker agents competing in ACPC.

In Poker, this is typically done by grouping different cards.

8.2.1 Lossless abstraction

Lossless abstraction of a game guarantees that an optimal strategy corresponds to an optimal strategy in the unabstracted game. In Poker, lossless abstraction is for example grouping some preflop cards. In the preflop phase, it doesn't matter if the player is dealt (A♣, K♢) or (A♡, K♠). Grouping cards is referred to as a **bucketing** - cards are grouped to **buckets**.

But how do we know that (A♣, K♢) and (A♡, K♠) are isomorphic preflop? How can we find all isomorphisms throughout the game? [16] show the *GameShrink* algorithm that finds all isomorphisms automatically. Using this algorithms, they were able to compute for the first time the exact Nash equilibrium in *Rhode Island Hold'em Poker*. Computing this game directly using linear programming yealds $91,224,226$ rows and columns. GameShrink reduces this to $1,190,443$ rows and $1,181,084$ columns. This linear program was solved using the interior-point barrier method of CPLEX in a week.

8.2.2 Lossy abstraction

Lossy abstraction of a game doesn't guarantee that an optimal strategy in the abstracted game will be optimal in the unabstracted game. So player's optimal strategy in the abstracted game can loose in the original game. Again, some grouping of preflop cards is loosy abstraction. If we group (A♣, A♢) and (2♣, 2♢), player can't distinguish these two states in original game.

Why would we want to use these abstractions? We can group much more information sets together, making the abstracted game much smaller. Sometimes we can't use only lossless abstraction, because the game is still too large to solve. This is the case of Hold'em Poker.

8.2.3 Imperfect recall in abstractions

Having a perfect recall abstraction, we don't forget any information. For example, we can use to 10 buckets in each round.

Other approach is to use more buckets on each round, but to forget the buckets from previous rounds. We can use 10 buckets for preflop 100 buckets on preflop, 1000 buckets on turn and finally $10,000$ buckets on river - and forget each round previous buckets. This graph has exactly the same size as having 10 buckets each round and remembering these buckets.

Creating this type of abstraction allows us to better capture the information for current round. Since this information seems more important than previous rounds, it could lead to better performing agents.

But forgetting previous buckets leads to game with imperfect recall. As we already know, there's not guarantee of existence of Nash equilibrium. Even worse, vanilla CFR minimization is ill-defined in this situation. This is because from one information set, there may be many actions leading to the same future information set. Fortunately, if we use public-chance sampling (so the sampled portion of the tree doesn't have this problem), CFR minimization is well-defined. Of course, there's still no guarantee of convergence, but this approach works well in practice.

Comparing perfect and imperfect recall abstractions shows that imperfect recall abstractions indeed perform better than their's (equally large) perfect counterparts. [43] [21].

8.2.4 Abstraction size and quality

Naturally, even if the abstraction is lossy, we want abstraction's strategy to perform well in the unabstracted game. In poker, choosing abstraction is crucial step in creating a good agent. One could think that the larger the abstraction, the better. Surprisingly, this is not that case even if the larger abstraction is *refinement* of the smaller one [42]. (informally, refinement allows player to make the same choices and possibly more). Let's show it on very nice example from [42]. Follows a simple, two players zero sum game in normal form

	a	b	c	d
A	7	2	8	0
B	7	10	5	6

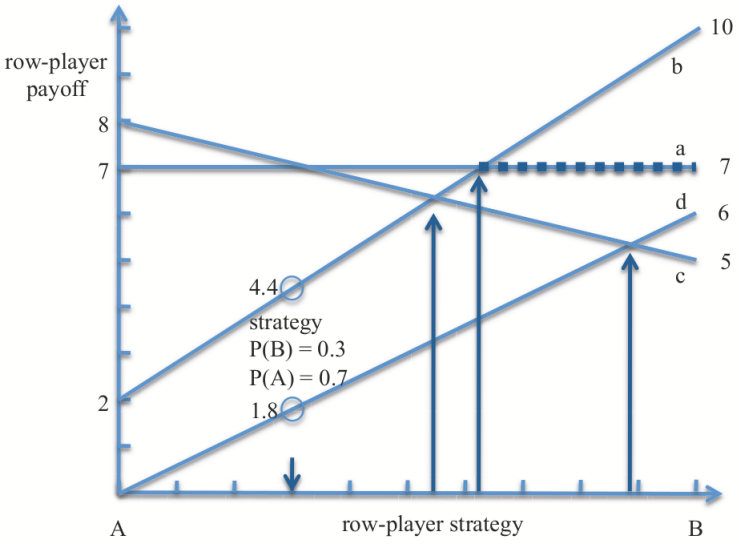


Figure 8.1: Visualization of the previous normal form game. [42]

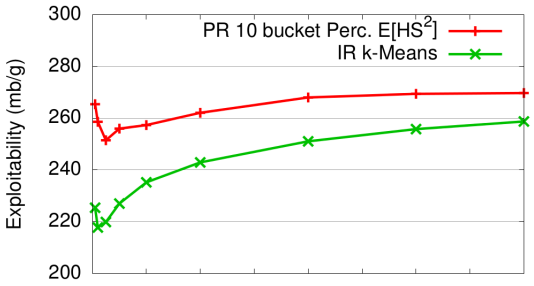
We see that the optimal strategy in the full game corresponds to the intersection of column player's strategies c and d. For the equilibrium, the guaranteed value for the row's player is slightly more than 5. Computing an equilibrium in the abstraction of this game, where the column player is allowed to use only a and b, yealds a set of equilibrium strategies (doted part on the figure). Adding another option c for the column player yealds new solution - the intersection of c and b. But if the row player plays this strategy in the unabstracted game, his worst-case utility is worse than for any doted strategy (that is, equilibrium if the column player was allowed only strategies a and b).

Pathology occurs for the row play as well. Suppose that the row player may play only B, the column player only c. In this case, row player's optimal strategy is to play B all the time. If we allow the row player to play A as well, his optimal strategy now becomes A. Clearly, in the unabstracted game, his guaranteed value for the strategy A is lower than for the strategy B. Why did that happen? Basically, having larger abstraction, the player can use strategies that seem valuable against opponent's abstraction. However, there may be strategies in the full game that counter these strategies, and the opponent is just not allowed to leverage that.

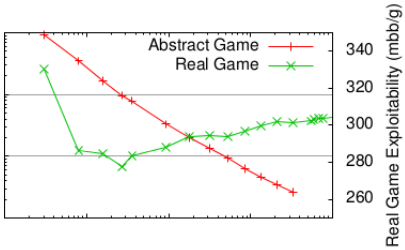
To sum it up, the strategy computed using the bigger abstraction can be beaten more than the strategy computed using the smaller abstraction.

8.2.5 Overfitting

Another drawback of dealing with abstraction is a possibility of overfitting. Computing a strategy in abstracted game converges to an equilibrium, or in other words, the exploitability gets lower and lower. But this holds just for the exploitability in the abstracted game. How will the strategy perform in the original game? Does it get better as well? As we will see later, it's tractable to compute the best response in the limit hold'em Poker. Having the possibility to compute the best response, we can check if the strategy gets better in the unabstracted game of limit hold'em Poker (see following figures)



(a) Perfect (red) and imperfect (green) recall abstractions. See that the strategy initially improves, but than steadily gets worse



(b) Green dots denote the exploitability in the unabstracted game, red in the abstraction. See that even though the exploitability in the abstraction goes to zero, this is not the case in the real game

I will show how to deal with both, overfitting and abstraction pathologies, later.

8.3 Card bucketing

Popular lossy abstraction technique is grouping private or public cards. This is called *bucketing* - cards are group to *buckets*. It's desirable to group cards that player should play similarly. But what cards should player play similarly?

8.3.1 E[HS]

E[HS] stands for "expected hand strength" - idea is to group hands with similar "strength". In this case, strength of hand is defined as a probability of beating opponents random hand. For example, player's odds with hand ($A\spadesuit, A\clubsuit$) against ($2\diamondsuit, 3\diamondsuit$) is 82.16%. If the flop ($4\diamondsuit, 5\diamondsuit, K\spadesuit$) is dealt, odds drop do 50.61%..

8.3.2 Hand strength distribution

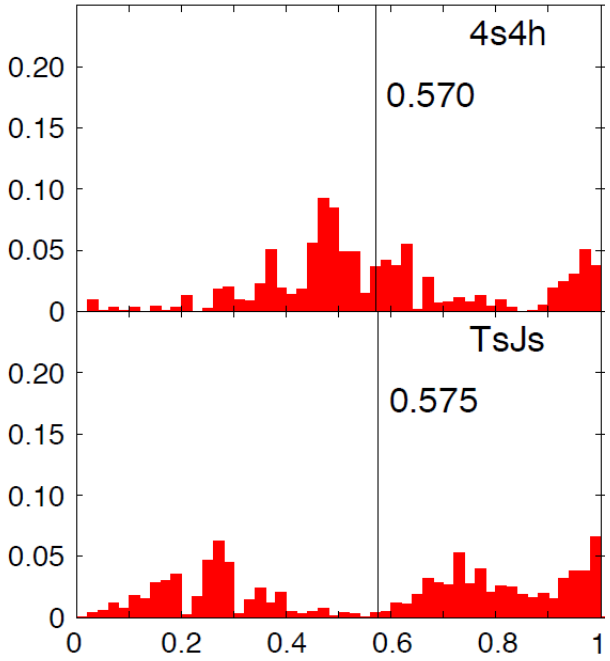


Figure 8.3: Two hands and the hand strength distribution for them. X-axis denotes the probability against random hands, Y-axis the corresponding number of such random hands (normalized). Note that E[HS] is the mean of this distribution and is displayed for both hands (0.570 and 0.575). We see that these hands have almost the same E[HS], but the distribution is quite different.

Grouping cards using only expected hands can group cards with different strength distribution (see figure above). Professional players believe that such cards should be played differently [37].

Note that this histogram shows only distribution over final hand strength, and doesn't take in account how the hand strength developed over time (intuitively, we want to group cards that develop similarly over time). But this histogram gives us still much more information than just an expected value.

Johanson et al. use different distances on this histogram to cluster similar hands. In this paper, they describe abstraction used by Hyperborean, one

8.3.3 Potential-aware automated abstraction

E[HS] nor hand strength distribution don't handle the development of the hand over the streets. Gilpin et al. [17] describe a technique that, simply stated, creates a histogram over future possible states (these states are actually discretized) and cards with similar histogram are grouped using Euclidian distance metric.

8.3.4 Accessing card bucketing

To translate from unabstracted game to abstracted one, agent needs to find appropriate bucket. Unfortunately, bucketing could be time-consuming. If it takes too much time to compute appropriate bucket, agent can't play the game in real-time. Straightforward workaround is to use some lookup table - bucket for any hand.

Naive implementation needs approximately $\binom{52}{2}\binom{50}{3}\binom{47}{1}\binom{46}{1} \approx 5.62 \times 10^{10}$. Using two bytes for each hand, this table would require more than 100 gigabytes to store. Of course, this is possible using current computers.

This table could be made much smaller - since some hands are equally strong, it's possible to exploit this symmetricity. Gilpin et al. [17] showed how this table could be reduced by a factor of ≈ 23, making the table much smaller.

8.4 Annual Computer Poker Competition 2012

8.4.1 Participants

Follows a list of participants of the no-limit tournament with brief descriptions. Descriptions are given by the authors and I just copied them from the competition website. As we see from these description, variety of approaches were used to develop these agents.

Azure Sky

- **Affiliation:** Azure Sky Research, Inc
- **Location:** Berkeley CA US
- **Technique:** SARSA trained neural nets, k-armed bandits, secret sauce.

dcubot

- **Affiliation:** School of Computing, Dublin City University
- **Location:** Dublin 9, Ireland
- **Technique:** The bot uses a structure like a Neural Net to generate its own actions. A hidden Markov model is used to interpret actions i.e. read an opponent's hand. The whole system is then trained by self-play. For any decision, the range of betting between a min-bet and all-in is divided into at most twelve sub-ranges. The structure then selects a fold,call, min-bet, all-in or one of these sub-ranges. If a sub-range is selected, the actual raise amount is drawn from a quadratic distribution between the end-points of the sub-range. The end-points of the sub-ranges are learnt using the same reinfrocement learning algorithm as the rest of the structure.

hugh

- **Affiliation:** Independent
- **Location:** NY, US Toronto, Ont, CA
- **Technique:** Ben (poker player and son) attempts to teach Stan (programmer and father) to play poker. Stan attempts to realize Ben's ideas in code.

  More specifically, pure strategies are utilized throughout. Play is based on range vs range ev calculations. PreFlop ranges are deduced by opponent modeling during play. Subsequent decisions are based a minmax search of the remaining game tree, coupled with some tactical considerations.

Hyperborean2pNL

- **Affiliation:** University of Alberta
- **Location:** Edmonton, Alberta, Canada
- **Technique:** Our 2-player no limit bot was built using a variant of Counterfactual Regret Minimization (CFR) ([3], [4]) applied to a specially designed betting abstraction of the game. Using an algorithm similar to the CFR algorithm, a different bet size is chosen for each betting sequence in the game ([1], [2]). The card abstraction used buckets hands and public cards together using imperfect recall, allowing for 18630 possible buckets on each of the flop, turn and river.

LittleRock

- **Affiliation:** Independent
- **Location:** Lismore, Australia
- **Technique:** LittleRock uses an external sampling monte carlo CFR approach with imperfect recall. Additional RAM was available for training the agent entered into this year's competition, which allowed for a more fine grained card abstraction, but the algorithm is otherwise largely unchanged. One last-minute addition this year is a no-limit agent.

  The no-limit agent has 4,491,849 information sets, the heads-up limit agent has 11,349,052 information sets and the limit 3-player agent has 47,574,530 information sets. In addition to card abstractions, the 3-player and no-limit agents also use a form of state abstraction to make the game size manageable.

Lucky7_12

- **Affiliation:** University of Maribor
- **Location:** Maribor, Slovenia
- **Technique:** We have developed a multi agent system that uses 8 strategies during gameplay. By identifying the state of the game, our system chooses a set of strategies that have proved most profitable against a set of training agents. The final decision of the system is made by averaging the decisions of the individual agents.

  The 8 agents included in our system are most rule-based agent. The rules for each individual agent were constructed using different knowledge bases (various match logs, expert knowledge, human observed play...) and different abstraction definitions for cards and actions. After a set of test matches were each agent dueled against the other agents in system, we determined that none of the included agents present an inferior or superior strategy (meaning each agent lost at least against one of the other agents and won at least one match).

Neo Poker Bot

- **Affiliation:** Independent
- **Location:** Spain
- **Technique:** Our range of computer players was developed to play against humans. The AI was trained on top poker rooms real money hand history logs. The AI logic employs different combinations of Neural networks, Regret Minimization and Gradient Search Equilibrium Approximation, Decision Trees, Recursive Search methods as well as expert algorithms from top players in different games of poker. Our computer players have been tested against humans and demonstrated great results over 100 mln hands. The AI was not optimized to play against computer players.

SartreNL

- **Affiliation:** University of Auckland
- **Location:** Auckland, New Zealand
- **Technique:** SartreNL uses a case-based approach to play No Limit Texas Hold'em. Hand history data from the previous years top agents are encoded into cases. When it is time for SartreNL to make a betting decision a case with the current game state information is created. The case-base is then searched for similar cases. The solution to past similar cases are then reused for the current situation.

Spewie Louie

- **Affiliation:** Georgetown University
- **Location:** Washington DC, USA
- **Technique:** The bot assumes bets can occur in: .25x, .4286x, .6666x, 1x, 1.5x, 4x, and 9x pot increments. Nodes in the tree contain: A hand range for each player, an "effectiveMatrix" that summarizes the tree below that point in the tree, and a "strategyMatrix" which is used by the "hero" of that node. Prior to the competition a collection of 24 Million matrices (1/2 strategy and 1/2 effective) were refined while simulating roughly 12.5 Million paths through the tree. This set of 24 Million matrices is then trimmed down to 770k (strategy only) matrices for the competition. Any decision not supported by this set of matrices is handled by an "on line" tree learned. During the learning process the set of effectiveMatrices and strategy matrices are stored in a ConcurrentHashMap. This gives the learning process good multi-thread behavior. Preflop hands are bucketed into 22 groups. Flop and Turn hands are bucketed into 8 groups. River hands are bucketed into 7 groups.

Tartanian5

- **Affiliation:** Carnegie Mellon University
- **Location:** Pittsburgh, PA, 15217, United States
- **Technique:** Tartanian5 plays a game-theoretic approximate Nash equilibrium strategy. First, it applies a potential-aware, perfect-recall, automated abstraction algorithm to group similar game states together and construct a smaller game that is strategically similar to the full game. In order to maintain a tractable number of possible betting sequences, it employs a discretized betting model, where only a small number of bet sizes are allowed at each game state. Approximate equilibrium strategies for both players are then computed using an improved version of Nesterov's excessive gap technique specialized for poker. To obtain the final strategies, we apply a purification procedure which rounds action probabilities to 0 or 1.

8.4.2 Results

	Round 0	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6	Round 7	Round 8	Round 9
hyperborean	586 ± 30	527 ± 28	530 ± 31	273 ± 23	223 ± 25	167 ± 27	167 ± 31	181 ± 37	174 ± 24	161 ± 36
tartanian5	597 ± 27	536 ± 27	509 ± 30	159 ± 30	97 ± 27	-7 ± 26	17 ± 29	4 ± 34	-76 ± 27	-161 ± 36
neo.poker.lab	534 ± 23	456 ± 23	424 ± 25	198 ± 23	86 ± 26	16 ± 28	-58 ± 31	-78 ± 32	-97 ± 23	–
little.rock	1116 ± 27	929 ± 28	907 ± 30	298 ± 26	128 ± 25	37 ± 27	-24 ± 35	-107 ± 35	–	–
sartre	112 ± 25	19 ± 25	-21 ± 29	56 ± 22	-57 ± 22	-97 ± 23	-101 ± 27	–	–	–
hugh	483 ± 18	459 ± 19	422 ± 22	71 ± 18	-49 ± 17	-117 ± 18	–	–	–	–
spewy.louie	355 ± 28	257 ± 29	204 ± 33	-230 ± 31	-427 ± 31	–	–	–	–	–
lucky7.12	-809 ± 57	-719 ± 47	-1009 ± 52	-826 ± 45	–	–	–	–	–	–
azure.sky	-196 ± 65	-1209 ± 60	-1966 ± 68	–	–	–	–	–	–	–
dcubot	-1097 ± 15	-1254 ± 16	–	–	–	–	–	–	–	–
uni.mb.poker	-1681 ± 47	–	–	–	–	–	–	–	–	–

Figure 8.4: Results - won/lost chips ($\pm$ variance)

We see that the winning agent uses techniques already described in this thesis - counterfactual regret minimization, regret matching, imperfect recall abstraction. The results presented in the table are "player vs player" - we can't see how good the agents are absolutely. In other words, we don't know the exploitability of these agents (best response).

8.5 Computing best response

To compute a best response, we need to traverse the game tree only once. As already stated, Texas hold'em poker has 10^{18} states, which makes this tasks intractable.

Exploitability is typically measured in milli-big-blinds per game (mbb/g), where a milli-big-blind is one one-thousandth of a big blind.

[22] show how to accelerate the computation and consequently computed best response for the limit version of Texas hold'em poker. There are actually three orthogonal ways they use to accelerate the computation, which I will very briefly describe

1. traversing a different type of tree
2. efficiently computing utilities in terminal nodes
3. game-specific isomorphisms to reduce the size of the tree
4. solving the task in parallel

Traversing a different type of tree Informally, **Public State Tree** is a game tree that an observer sees. Because he can't see any private cards, all states with the same betting history and public cards are indistinguishable. Traversing this type of tree allows to reuse the queries to the opponent's strategy. We will see this tree once more in the chapter 9.

Efficiently computing utilities in terminal nodes Given a terminal node in the public state tree, we need to compute utilities for each of our information sets grouped in this node (for example, the information set in which we hold $Q\spadesuit, Q\clubsuit$). This terminal node contains opponent's information sets with corresponding reach probabilities (for example 50% $K\spadesuit K\clubsuit$, 50% $J\spadesuit J\clubsuit$).

Given our information set, we can compute the value against the opponents distribution Doing this for each our information sets gives $O(n^2)$ operations. Leveraging the fact that poker hands form an ordering, we can first sort the hands by strength and than compute the values for all our information sets lineary. This gives us $O(nlog(n))$.

Solving the task in parallel Raching a public tree node with corresponding reach probabilities, the computation for all of its children can be done in parallel. For example, we can do this splits on flop - we compute the reach probabilities for the flop and then compute all non-isomorphic flops in parallel.

8.5.1 Results

Combining all these enhancements, [22] were able to compute the best response in the limit Hold'em poker in a day.

8.6 Performing well in the unabstracted game

One way to overcome both, overfitting and abstraction pathologies is to create an abstraction only for one of the players. [42] show that if only one of the players is abstracted, the refined abstraction indeed guarantees at least the same game value in the original game. This is not surprising, since the unabstracted player "punishes" the abstracted one in the original game.

But the very reason for creating the abstraction is that the original game is too large. Fortunately, we can use the best-response as the opponent.

8.6.1 CFR-BR

Stands for Counter Factual Regret - Best Response. In CFR minimization, both players were using some regret minimizing algorithm (such as regret matching) to minimize their overall regret, consequently finding an ϵ-equilibrium.

In CFR-BR, one player uses CFR minimization, while the opponent always play the best response to his current strategy. First, notice that to compute the best response, we need to know the current strategy (in contract to CFR, where both players could compute their strategies simultaneously). This means that one cannot use best response strategy for both players.

The convergence follows from the fact that one player is regret minimizing and the second player has no positive regret on every iteration [21].

8.6.2 Hybrid agent

While using CFR-BR is possible, computing best response on each iteration is still nontrivial.. [21] use a variant of Monte Carlo sampling and sample a public chance event in the beginning of the game.

This way, the divide the game tree to the trunk (part of the tree prior to the sampled action) and subgames. The trick is to compute the best response on each iteration only in the currently sampled subgame, and use Counter Factual regret minimization in the trunk. This hybrid agent minimizes overall regret [21]. Public chance event dividing the trunk and subgames can be arbitrary, it's a matter of balancing time and memory requirements. [21]. achieved the best results by using 2-round trunk - flop and turn cards are sampled on each iteration.

8.6.3 Best known strategy

The exploitability of Hyperborean2011.IRO (participant of the ACPC 2011), computed using the best response computation described above, was $104.410 mbb/g$. The abstraction had 5.8 billion information sets, and the total size was about $20GB$. Using the Hybrid CFR-BR algorithm on the same abstraction, [21] computed a strategy that is exploitable only by $41.199 mbb/g$ - this is the least exploitable strategy known for heads-up limit Texas Hold'em Poker.

8.7 Annual Computer Poker Competition 2013

The results of ACPC 2013 were announced only few days before the deadline for this thesis. I'm happy to announce that our team finished on the 4th place out of 13 competitors.

9. No-limit betting

In this chapter, I show an interesting result regarding no-limit betting.

9.1 Actions count

In contract to limit betting, player is allowed to bet any amount up to the player's stack. This makes the game tree much larger. In limit version, the number of actions (outcoming edges) is low - *check/call*, *fold* and *bet*.

On the other hand, if player's stack is 10000$, the big blind is 1$ and he can bet any integer amount, the number of actions can be 10000 (of course, this depends and the particular node, if the node corresponds to the situation where both players already made some bets, there will be fewer actions).

Fore example in the no-limit poker played in the *Annual Computer Poker Competition* 2013, there were up to 20 000 bet sizes available in every information set, and the game consequently contains 6.3×10^{164} game states [20].

9.2 Mixing actions

In games with no hidden information, players don't need to mix their strategies to play optimally - in every game node, they just select the one, best action (best response). This is not the case in games with imperfect information, where player may need to mix his actions. But how many actions does he need to mix?

Imagine a game where player receives always one (the same) card (for example always ace). Opponent knows which card he was dealt. This makes it a game with perfect information and player can select just one action in any information set. What if we add another card?

Does the player now need to mix all his actions? If there's only one card, player can use only one action - having two cards, can he use only two actions? Or are there situations where he needs to mix all available actions?

9.3 Public State Tree

Informally, public state tree is how the game looks like to an observer. Observer sees the actions player make as well as all public cards, but he can't see private cards of any player. Based on the strategies, we can compute distribution over grouped nodes in the public state tree using Bayes' rule.

*We call a partition of the histories, $\mathcal{P}$, a **public partition** and $P \in \mathcal{P}$ a **public state** if [22]*

- *No two histories in the same information set are in different public states*
- *Two histories in different public states have no descendants in the same public state*
- *No public state contains both terminal and non-terminal histories*

For this public state tree, we also define [33]:

- *A set of actions available in every $P \in \mathcal{P}$*

$$A(P) := \cup_{I \in P} A(I)$$

- *Acting player in P*

$$p(P) := p(I) \text{ for any } I \in P$$

- $\lambda(P)$ *- information sets of player $p(P)$ grouped in this public node*
- $\nu(P, I)$ *- Probability of being in information set $I \in P$, conditional on $P \in \mathcal{P}$ being reached (consistent with σ)*
- *Probability measure over $A(PI)$*

$$\gamma(P, a) = \sum_{I \in P} \nu(P, I)\beta(I, a)$$

- *$P_a \in \mathcal{P}$, public state that follows $P \in \mathcal{P}$ if action $a \in A(P)$ is taken.*

$$(h, a) \in P_a \in \mathcal{P} \iff (h) \in P \in \mathcal{P}$$

- ***Counterfactual information set*** *CI^a corresponding to $I \in \lambda(P)$. We refer to these information sets as counterfactual, because they would be information sets if the player $p(P)$ was to play again (and not an opponent).*

$$(h) \in I \iff (h, a) \in CI^a$$

For counterfactual information sets, we also analogically define $\lambda_c(P_a)$, $\nu_c(P_a, CI^a)$, and μ_c.

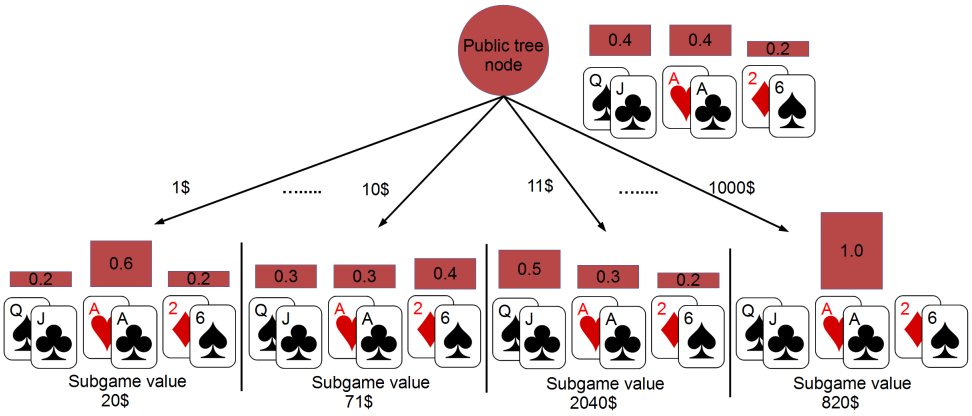


Figure 9.1: Public tree with displayed ranges.

Public State Tree and Poker

Follows a small portion of public state tree of no-limit Hold'em Poker. Note that Poker community usually refers to $\nu(P, I)$ as a *range*. In other words, range is distribution over player's cards given the public tree node.

9.4 Changing the strategies

Given a public game tree of a Nash equilibrium If we use just a subset of bets without changing any range, opponent's best response remains the same This also means that the subgame values don't change If we don't change the overall game value we gain, our strategy remains best response as well Both players are playing best response, so the strategy pair forms a new Nash equilibrium How to find some small subset?

9.5 Outline of approach

Let's suppose there's an sequential equilibrium (σ, μ) using more than $|\lambda(P)|$ actions in any $I \in P$ (if there's no such equilibrium, we are done). We create new assessment (σ', μ) that differs only in information sets in P, so that this assessment satisfies:

1. sequential rationality
2. consistency
3. $|$**actions used by new strategy in P**$| \leq |\lambda(P)|$

So we get a new sequential equilibrium, using no more than $|\lambda(P)|$ actions in every $I \in P$. Iteratively, we take this new equilibrium and if there's another $I \in P$ that uses too many actions, we just repeat steps above. Finally, since the game is finite, we get Nash equilibrium using no more than $|\lambda(P)|$ actions in every information set in P.

9.6 Proof

We denote $\beta, \mu, \nu, \pi, \gamma$ of the new strategy σ' as $\beta', \mu', \nu', \pi', \gamma'$.

Given an assessment (σ, μ), we find P that violates action bound. Now we want to compute new strategy profile σ', but since we don't change beliefs, (σ', μ) must be *consistent*. First step is to show that.

Lemma 3. If for all P_a, for all $CI^a \in \lambda_c(P_a)$:
$\nu_c = \nu'_c$, then (σ', μ) is consistent [33]:

With this result in mind, we write down some simple equations, where each variable x_i correspond to $\gamma'(P, a_i)$ so that $\nu_c = \nu'_c$.

$$\begin{pmatrix} \sum_i \nu_c(P_{a_i}, CI_1^{a_i} \in \lambda_c(P_{a_i}))\, x_i \\ \sum_i \nu_c(P_{a_i}, CI_2^{a_i} \in \lambda_c(P_{a_i}))\, x_i \\ \vdots \\ \sum_i \nu_c(P_{a_i}, CI_{|\lambda(P)|}^{a_i} \in \lambda_c(P_{a_i}))\, x_i \end{pmatrix} = \begin{pmatrix} \nu(P, I_1) \\ \nu(P, I_2) \\ \vdots \\ \nu(P, I_{|\lambda(P)|}) \end{pmatrix}$$
$$x_i \geq 0 \ \forall i \tag{9.-1}$$

See that for any solution, $\sum_i x_i = 1$. [33]:

9.6.1 New strategy

Because x_i correspond to new $\gamma'(PI, a_i)$, we set $\beta'(I, a)$ to:

$$\beta'(I, a_i) = \frac{\nu_c(P_{a_i}, CI^{a_i}) x_i}{\nu(P, I)} \tag{9.-1}$$

9.6.2 Strategy properties

1. β' is valid distribution: $^{(*)}$

$$\sum_{a_i \in A(I)} \beta'(I, a_i) = 1$$

2. Beliefs remain consistent: $^{(*)}$

$$\forall a \in A(P), \forall I \in \lambda(P) : \ \nu'_c(P_a, CI^{a_i}) = \nu_c(P_a, CI^{a_i})$$

So any solution to (9.6) gives us new strategy σ', so that assessment $((\sigma'_i, \sigma_{-i}), \mu)$ remains consistent. Since beliefs remain unchanged, we know that all players except of i are *sequentially rational*.

9.6.3 Sequential rationality

To satisfy sequential rationality for player $p(P)$, we simply maximize his expected value:

$$f(x) = \sum_{a_i \in A(P)} x_{a_i} O_{p(P)}(\sigma', \mu | (P_{a_i})) \tag{9.-1}$$

9.6.4 Action elimination

Maximizing function (9.6.3) over conditions (9.6) gives us new sequential equilibrium. But both, the conditions and function are linear! Thus there must be some optimal *basic* solution, using no more than $rank(A)$ non-zero variables [**?**]. Finally, because there are only $|\lambda(P)|$ rows in (9.6), this concludes our proof.

10. Conclusion

This thesis described the state of the art algorithms and techniques used to solve large extensive form games, namely Poker. I also presented an interesting result regarding no-limit betting (which is not limited to Poker).

10.1 Future work

Looking at the Annual Computer Poker Competition, we see more and more sophisticated participants every year. But poker strategies have still long way to go, particularly in the no-limit version.

We hope to continue our work on game theory, imperfect information games etc. We already have some new ideas and approaches in our minds. Finally, this was the first time we participated in the ACPC, and we definitely plan to compete next year with much better agent. Our goal is to beat all ACPC 2013 agents using our new program and use that program in ACPC 2014.

Bibliography

[1] Bellhouse, D. The problem of waldegrave. *Electronic Journal for the History of Probability and Statistics 3* (2007), 1–12.

[2] Billings, D., Papp, D., Schaeffer, J., and Szafron, D. Poker as a testbed for ai research. *Advances in Artificial Intelligence* (1998), 228–238.

[3] Border, K. C. *Fixed point theorems with applications to economics and game theory*. Cambridge university press, 1989.

[4] Brandt, F., Fischer, F., and Harrenstein, P. On the rate of convergence of fictitious play. In *Algorithmic Game Theory*. Springer, 2010, pp. 102–113.

[5] Chen, B., and Ankenman, J. *The mathematics of poker*. ConJelCo LLC, 2006.

[6] Chen, X., and Deng, X. Settling the complexity of two-player nash equilibrium. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on* (2006), IEEE, pp. 261–272.

[7] computerpokercompetition. computerpokercompetition, 2013.

[8] Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. The complexity of computing a nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing* (2006), ACM, pp. 71–78.

[9] Doe, R. The second man-machine poker competition, 2008.

[10] en.chessbase.com. en.chessbase.com, 2011.

[11] Ferber, J. *Multi-agent systems: an introduction to distributed artificial intelligence*, vol. 1. Addison-Wesley Reading, 1999.

[12] FUDENBERG, D. A. *The theory of learning in games*, vol. 2. MIT press, 1998.

[13] Ganzfried, S., and Sandholm, T. Computing an approximate jam/fold equilibrium for 3-player no-limit texas hold'em tournaments. In *Proceedings of the 7th international joint conference on Autonomous agents and multi-agent systems-Volume 2* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 919–925.

[14] Gilboa, I., and Zemel, E. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior 1*, 1 (1989), 80–93.

[15] Gilpin, A., Hoda, S., Pena, J., and Sandholm, T. Gradient-based algorithms for finding nash equilibria in extensive form games. In *Internet and Network Economics*. Springer, 2007, pp. 57–69.

[16] Gilpin, A., and Sandholm, T. Lossless abstraction of imperfect information games. *Journal of the ACM (JACM) 54*, 5 (2007), 25.

[17] Gilpin, A., Sandholm, T., and Sorensen, T. B. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE* (2007), vol. 22, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 50.

[18] Goldberg, P. W., and Papadimitriou, C. H. Reducibility among equilibrium problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing* (2006), ACM, pp. 61–70.

[19] HoldemResources. Holdem resources poker tools, 2013.

[20] Johanson, M. Measuring the size of large no-limit poker games. *arXiv preprint arXiv:1302.7008* (2013).

[21] Johanson, M., Bard, N., Burch, N., and Bowling, M. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)* (2012).

[22] Johanson, M., Waugh, K., Bowling, M., and Zinkevich, M. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One* (2011), AAAI Press, pp. 258–265.

[23] Kan, M. H. *Postgame analysis of poker decisions*. PhD thesis, University of Alberta, 2007.

[24] Kass, S. Rock paper scissors spock lizard, 2013.

[25] Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. Monte carlo sampling for regret minimization in extensive games. *Advances in Neural Information Processing Systems 22* (2009), 1078–1086.

[26] Myerson, R. B. *Game theory: analysis of conflict*. Harvard University Press, 1997.

[27] Nash, J. Non-cooperative games. *Annals of mathematics 54*, 2 (1951), 286–295.

[28] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. *Algorithmic game theory*. Cambridge University Press, 2007.

[29] Osborne, M. J., and Rubinstein, A. *A course in game theory*. MIT press, 1994.

[30] Papadimitriou, C. H. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences 48*, 3 (1994), 498–532.

[31] Pokerstars. www.thedailybeast.com, 2011.

[32] Robinson, J. An iterative method of solving a game. *The Annals of Mathematics 54*, 2 (1951), 296–301.

[33] Schmid, M., and Moravcik, M. Equilibrium's action bound in extensive form games with many actions. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence* (2013).

[34] Shapley, L. S., et al. Some topics in two-person games. *Advances in game theory 52* (1964), 1–29.

[35] Sinervo, B., and Lively, C. M. The rock-paper-scissors game and the evolution of alternative male strategies. *Nature 380*, 6571 (1996), 240–243.

[36] Sklansky, D. *The theory of poker*. Two Plus Two Publishing LLC, 1999.

[37] Sklansky, D., and Malmuth, M. *Hold'em poker for advanced players*. Two Plus Two Pub, 1999.

[38] Smith, J. M., and Price, G. lhe logic of animal conflict. *Nature 246* (1973), 15.

[39] Szafron, D., Gibson, R., and Sturtevant, N. A parameterized family of equilibrium profiles for three-player kuhn poker. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (2013), International Foundation for Autonomous Agents and Multiagent Systems, pp. 247–254.

[40] texasholdempokerstars. texasholdempokerstars, 2013.

[41] thedailybeast. www.thedailybeast.com, 2005.

[42] Waugh, K., Schnizlein, D., Bowling, M., and Szafron, D. Abstraction pathologies in extensive games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (2009), International Foundation for Autonomous Agents and Multiagent Systems, pp. 781–788.

[43] Waugh, K., Zinkevich, M., Johanson, M., Kan, M., Schnizlein, D., and Bowling, M. A practical use of imperfect recall. In *Proceedings of the Eighth Symposium on Abstraction, Reformulation and Approximation (SARA)* (2009), pp. 175–182.

[44] Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. In *Advances in neural information processing systems* (2007), pp. 1729–1736.

List of Tables

List of Abbreviations

Attachments