Evaluation of Equity Models for Tournament Poker

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Intelligente Systeme

eingereicht von

Helmuth Melcher, Bakk.techn.
Matrikelnummer 0125557

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao. Univ.-Prof. Mag.rer.nat. Dipl.-Ing. Dr.techn. Rudolf Freund

Wien, 27. November 2015

Helmuth Melcher	Rudolf Freund

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Evaluation of Equity Models for Tournament Poker

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Intelligent Systems

by

Helmuth Melcher, Bakk.techn.
Registration Number 0125557

to the Faculty of Informatics

at the Vienna University of Technology

Advisor: Ao. Univ.-Prof. Mag.rer.nat. Dipl.-Ing. Dr.techn. Rudolf Freund

Vienna, 27th November, 2015

Helmuth Melcher | Rudolf Freund

Erklärung zur Verfassung der Arbeit

Helmuth Melcher, Bakk.techn.
Brunnengasse 42/3/302, 1160 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. November 2015

Helmuth Melcher

Acknowledgements

First and foremost, my deepest gratitude goes to my good friend **Paul Schwanzer** for pushing me to finish my Masters program after a university hiatus of more than seven years. I would not have started this work without him.

I am also extremely thankful to my advisor **Dr. Rudolf Freund** for his understanding supervision and great help in making this work possible, despite the tight time schedule. Acknowledgements also to **Andrew**, **Carmen** and **Peter**, for taking the time to proof read and provide me with feedback.

Kurzfassung

Turnier-Poker ist ein beliebtes stochastisches Spiel mit imperfekter Information. Das Spiel bewegt sich abhängig von den Aktionen der Spieler zufällig in einer Menge von Turnierzuständen. Heuristische *Equity Modelle* sind ein häufig genutztes Werkzeug um die erwarteten Auszahlungen der Spieler für einen gewählten Turnierzustand anhand der Chipstände zu approximieren. Trotz der Beliebtheit dieser Modelle, und einer Auswahl von dedizierter Software um sie auf Spielsituationen anzuwenden, gibt es kaum öffentlich verfügbare Evaluierungen der verwendeten Methoden.

Als Teil dieser Arbeit wird durch Anwendung von aktuellen Techniken zur Lösung von Spielen mit imperfekter Information ein ϵ-Nash Gleichgewicht für eine umfangreiche Abstraktion eines Poker-Turnierspiels berechnet. Die Auszahlungswerte und Strategien dieser Lösung werden im Folgenden als Referenz verwendet, um die Genauigkeit der heuristischen *Equity Modelle* zu untersuchen.

Die bereits bekannte Tendenz des Malmuth-Harville Standardmodells zur Unterschätzung der Auszahlungen für Spieler mit großen Chipständen wird als Teil dieser Evaluierung bestätigt, und wird von dem neueren Roberts Modell in einem geringeren Ausmaß geteilt.

Von einer der zugrunde liegenden Annahmen der untersuchten Heuristiken, die Gewinnwahrscheinlichkeiten der Spieler wären proportional zu ihren Chipständen, wurden in der berechneten Spielabstraktion bemerkenswerte Abweichungen gefunden. Für die ebenfalls betrachtete Suchtechnik *Future Game Simulation (FGS)* konnte eine deutliche Verbesserung gegenüber den Standardmodellen festgestellt werden.

Abstract

Tournament poker is a popular stochastic game with imperfect information, the game transitions probabilistically between tournament states dependent on the players' actions. Several heuristic *equity models* are widely employed by human players to estimate the expected payoffs for tournament states. Despite the popularity of these models and the availability of dedicated software for applying them, little public research is available regarding their accuracy.

In this work, we calculate an ϵ-Nash equilibrium for a large tournament game instance by applying state-of-the-art techniques for solving games of imperfect information, and use the results of this game to evaluate the performance of the heuristic approaches.

We find that the de facto standard Malmuth-Harville heuristic and the model proposed by Roberts display similar accuracy in our game. The known tendency of Malmuth-Harville to underestimate payoffs for large chip stacks is shared by Roberts, although to a smaller degree.

The common underlying assumption of winning probabilities being proportional to the chip stacks does not hold true in our game setting. We also evaluate an adversarial search technique called Future Game Simulation (FGS), and find that it provides a significant improvement over the standard heuristics.

Contents

CHAPTER 1

Introduction

The game of poker offers interesting challenges for research in the fields of artificial intelligence and game theory that are not present in games like chess or backgammon. As a game of chance with imperfect information, agents need to deal with uncertainty and make deductions about hidden information.

As John von Neumann, one of the authors of [VNM44] and a pioneer in the field of game theory stated: “Real life consists of bluffing, of little tactics of deception, of asking yourself what is the other man going to think I mean to do. And that is what games are about in my theory.”[Pou92]

Poker has traditionally also been a game played for considerable amounts of money. With the rise of online poker in the early 2000s, the number of players dedicated to studying the game has increased substantially.

With significant interest from recreational and professional human players, along with a sizable market, the segment of poker tools and instructional services has flourished in recent years. Especially for online play, a wide variety of poker tools are available to help players with learning and improving their understanding of the game. These tools range from extremely popular database software and heads-up displays to analytical game theory software.

In this latter category of poker related game theory software, tournament poker - specifically single table tournaments - have led the development. The endgame in these tournaments allows some powerful abstractions which make calculation of hands considerably less complex than their *cash game* counterparts.

Correct play in tournament late game is often surprisingly counter-intuitive because of the non-linearity between chips stacks and expected payoffs. Players realized that studying models to better understand the game structure is essential for playing competently in this format.

With the underlying calculations being far too complex to perform manually, dedicated software tools became popular among players. The first generation of tournament analysis tools was released around ten years ago and provided a comparatively simple application of tournament equity models by today's standards.

Partly driven by the Annual Computer Poker Competition [ACP], most academic research focused on the *cash game* variants of poker and considerable progress has been made in this area during recent years. Early in 2015, the poker research group at the University of Alberta "essentially solved" the Fixed Limit Hold'em variant for two players.[BBJT15] This marked the first time that a poker variant somewhat popular among human players was solved to a high degree of accuracy using only lossless abstractions. Later in the same year, a team at Carnegie Mellon University organized the "Brains vs. AI" [Gan15] challenge match between their two player No Limit Hold'em agent "Claudico" and some of the best human players in this game format. Both research teams received mainstream media attention for their work.

While single table tournaments are some of the most analyzed games in terms of commercially available software, they received little attention in terms of academic research. A notable exception are the papers about 3-player jam/fold games by Ganzfried and Sandholm [GS08] [GS09] that we will use as a basis for this work. Outside of loose collaboration by some of the authors of commercial software, very little analysis of the heuristics used in these games has been published. This is especially surprising as Malmuth-Harville, the de-facto standard heuristic for tournament equity calculations, was actually published in the context of horse racing in 1973 [Har73].

1.1 Texas Hold'em Poker

We will first give a short introduction to the game rules and general game structure of Texas Hold'em. Like most poker variants, Texas Hold'em is a game of chance with imperfect information. It contains both stochastic elements (dealing cards from a shuffled deck) and hidden information (private *hole cards* visible only to a single player).

1.1.1 Game Overview

Texas Hold'em is a Poker variant with 4 betting rounds. At the beginning of a hand blinds and antes are posted and each player receives two private cards from a 52 card deck. These private cards are called *hole cards*, they remain hidden to all other players and are only revealed if there is a *showdown* at the end of the hand.

After the first betting round, public *community cards* are dealt on the *Flop* (3 cards), *Turn* (1 card) and *River* (1 card), each of these steps is followed by another betting round. After the final round of betting all remaining players see a *showdown* and compare their hands to determine the winner of the accumulated pot.

1.1.2 Positions, Blinds and Antes

The player positions during a poker hand are determined by a tag called the *dealer button* or Button (BU). It is placed in front of a random player at the start of the game and moves clockwise to the next player after every hand.

Before the hole cards are dealt, the two players in clockwise direction following the BU are the Small Blind (SB) and Big Blind (BB) positions. The players in these positions are required to post blind bets, usually with $BB = 2 \times SB$. Optionally all players have to post an additional *Ante*.

ID	Position	Preflop	Postflop
EP1	Early Position 1	1.	3.
EP2	Early Position 2	2.	4.
MP1	Middle Position 1	3.	5.
MP2	Middle Position 2	4.	6.
HJ	Hijack	5.	7.
CO	Cut-Off	6.	8.
BU	Button	7.	9.
SB	Small Blind	8.	1.
BB	Big Blind	9.	2.

Table 1.1: Positions and order to act Pre- and Postflop.

1.1.3 Hole Cards and Annotations

After *blinds* and *antes* are posted, each player is dealt two private hole cards from a 52 card deck. Single cards are usually annotated by a number or first letter representing the rank (2 . . . 9, Ten, Jack, Queen, King, Ace), followed by the suit, e.g.: $T\clubsuit$ or Tc for the Ten of Clubs.

During the *Preflop* betting round there are no community cards, so the $52C2 = 1326$ hole card combinations can be divided into 169 classes of strategically equivalent hands. For instance, the hole card combinations of $A\heartsuit K\diamondsuit$ and $A\clubsuit K\spadesuit$ are strategically equivalent on the first street.

1.1.4 Betting Rounds

During each betting round, if there has been no previous bet in the current betting round, the active player can either place a bet, or check to the next player. If there has been a previous bet, players can either call by matching the current bet, raise the bet, or surrender the chance to win the pot by folding their *hole cards*.

If a player uses all of their remaining chips to bet, raise or call, the player is declared *all-in*. Players who are *all-in* remain active in the hand without matching further bets, but they can only compete for the part of the pot that they matched. Any additional bets that were unmatched by some of the *all-in* players are separated into *side pots* and contested only among the players who fully matched the bets.

1.1.5 Hand Scoring

To score hands at *showdown*, each player remaining active in the hand combines his two private hole cards with the five public community cards to build the best possible five card hand. Table 1.2 gives a brief overview of the different ranks.

The pot is then awarded to the player with the highest five card hand. If multiple players have the best hand of the exact same rank, this is called a *split pot* and the pot is distributed equally among the tied players in this case.

Name	**Example**				
Royal Flush	A♥	K♥	Q♥	J♥	T♥
Straight Flush	9♣	8♣	7♣	6♣	5♣
Four of a Kind	7♠	7♥	7♦	7♣	K♠
Full House	A♥	A♦	A♣	2♠	2♣
Flush	K♠	9♠	8♠	7♠	4♠
Straight	K♥	Q♦	J♣	T♣	9♥
Three of a Kind	Q♠	Q♥	Q♦	7♦	5♠
Two Pair	9♥	9♣	3♠	3♦	K♣
One Pair	5♠	5♦	K♣	9♥	8♠
High Card	A♥	T♣	9♣	7♠	2♥

Table 1.2: Five card hand ranks.

1.2 Tournament Poker

Poker games can be divided into the categories of *cash* or *tournament* games. In a *cash game*, the poker chips have a fixed monetary value and players are free to quit the game at any point and exchange their chips into cash. In this scenario, the game payoffs correspond directly to the chip counts and are known at the end of every hand played. As the chips in cash games are directly convertible, there is no strategical difference between optimizing chip count and optimizing payoffs.

Tournament poker is different, in the sense that tournament chips do not have any direct monetary value during the tournament, so they can not be converted back into cash. The payoffs for a tournament game are defined by a payout structure $R = (r_1, r_2, \dots, r_n)$ with $r_i \geq r_{i+1}$, where r_i is the prize for the player finishing the tournament in place i.

Upon entering a tournament, players receive a specified amount of chips and players are eliminated from the tournament if they finish a hand with zero chips. The players receive their payoff according to the payout structure R when they are are either eliminated, or there is only one remaining player left in the tournament.

For the remainder of this work, we will focus on a special subset of tournaments with a relatively small number of entrants called Sit and Go (SNG) and more specifically Single Table Tournament (STT).

STTs are a very popular tournament format in online poker and typically offered with tournaments starting at 2, 6 or 9 players. The term SNG refers to the fact that these games are usually offered "on demand" without a fixed time schedule. Players can register at any time and a new game starts as soon as the target number of players is reached.

The main additional challenge for tournaments, compared to cash games, is that the expected payoffs at the end of a tournament hand are not generally known or trivial to estimate.

To demonstrate the general problem of estimating expected payoffs in tournament play, consider an example tournament with a prize structure of $R = (50\$, 30\$, 20\$)$ to be awarded to the players finishing in places $1 \ldots 3$; players eliminated before this receive no prize.

With four remaining players having chip stacks proportional to $x \propto (36, 4, 3, 2)$, player i_1 controls $36/45 = 80\%$ of the chips. The expected payoff for i_1 can clearly not exceed the 50% of the prize pool for finishing in first place. On the flip side, the remaining three players only control a total of $(4 + 3 + 2)/45 = 20\%$ of the chips and have a combined expected payoff of the remaining $> 50\%$ of the prize pool.

It is obvious from this example that expected payoffs are not proportional to chip stacks. This means that there can be considerable discrepancies between strategies that maximize the expected chip count, and strategies maximizing the expected payoff.

Being able to estimate the expected payoffs for various chip configurations is an essential prerequisite for making correct game decisions in poker tournaments. *Equity models* are heuristics used to estimate payoffs in these situations, usually based only on the prize structure of the tournament and the players' chip stacks.

1.3 Overview

We provide a detailed description of the popular *equity models* in chapter 2, and discuss state-of-the-art techniques for solving large stochastic games with imperfect information in chapters 3 and 4.

In chapter 5, we explore the complexity of various game abstractions and select an abstract tournament game setting that can be solved on our available hardware. Chapter 6 provides a brief discussion of the methods and parameters used for our calculations.

In the remainder of the work, we use the solution of this tournament game to evaluate heuristics introduced in chapter 2. We compare the accuracy of payoff estimates in chapter 7, using the expected payoff values in the tournament solution as a reference. The performance of strategy profiles based on the *equity models* is then evaluated in chapter 8 against both against a theoretical worst case “nemesis” opponent, and against the strategy profile calculated for the tournament solution. Chapter 9 concludes the work and provides a summary of our findings.

CHAPTER 2

Equity Models

In order to analyze poker tournaments at an individual hand level, without calculating the entire tournament, heuristics are commonly used to approximate the expected payoffs for players. These heuristics are called *equity models* in the poker community, with the most popular one being the Malmuth-Harville model, also known as the Independent Chip Model (ICM).

These models are typically much too complex for manual calculation. A variety of free and commercial software tools and websites are available to apply these models to game situations. Virtually, all of the software packages use Malmuth-Harville as the default model. The only popular alternative in widespread use is an adversarial search extention "FGS" that works on top of the Malmuth-Harville model. We will discuss this algorithm in detail in section 4.2.

The models discussed in this section approximate the expected payoffs by estimating the finishing distribution of players based on their chip stacks.

For a tournament with n players, the finishing distribution matrix P is an $\mathbb{R}^{n \times n}$ matrix where $p_{i,j}$ is the probability that player i finishes in place j. As these are probabilities, all values in the matrix are non-negative and each row and column has a sum of 1. The expected payoffs for each player can then be calculated by multiplying the player rows of P with the payout vector R.

In a similar fashion to [CA06, ch.27], we will use the following notation:

- x_i is the chip stack of a player $i \in N$
- $X_{i,j}$ is the event of player i finishing the tournament in place j
- $v(x, R)$ is the vector of payoff estimates for a tournament state with chip stacks x and a tournament payout structure R

2.1 Malmuth-Harville

The Malmuth-Harville model [Har73][CA06][GS08] is based on the following two assumptions:

1. The probability of a player i finishing in first place is proportional to the player's chip stack x_i.
$$Pr(X_{i,1}) \propto x_i \tag{2.1}$$
2. The conditional probability of player i finishing in place j, given that players $a, b, \ldots$ finish in the first $(j-1)$ places, is proportional to the chip stacks of the remaining players.

$$Pr(X_{i,j} | X_{a,1} \wedge X_{b,2} \wedge \ldots) = \frac{x_i}{\sum_{k \notin \{a,b,\ldots\}} x_k} \tag{2.2}$$

This model is commonly implemented by simply enumerating all possible elimination sequences of the players. In this case, the runtime grows factorially with the number of players $O(n!)$ and is only practical for very low n.

2.1.1 Example

The finishing distribution matrix as estimated by the Malmuth-Harville heuristic P_H for chip stacks of $x = [2, 5, 8, 13, 17]$ is shown below. The estimated payoffs υ_H for a payout structure of $R = [50, 30, 20, 0, 0]$ are also included.

$$P_H(x) = \begin{bmatrix} 0.044 & 0.060 & 0.093 & 0.180 & 0.622 \\ 0.111 & 0.142 & 0.201 & 0.331 & 0.214 \\ 0.178 & 0.211 & 0.264 & 0.246 & 0.102 \\ 0.289 & 0.287 & 0.239 & 0.145 & 0.040 \\ 0.378 & 0.300 & 0.202 & 0.098 & 0.022 \end{bmatrix}$$

$$\upsilon_H(x, R) = P_H(x) \times R^T = \begin{bmatrix} 5.89 & 13.84 & 20.48 & 27.86 & 31.94 \end{bmatrix}$$

2.1.2 Improved Implementation

This brings us to the first contribution of this work, an optimized implementation of the Malmuth-Harville heuristic which calculates the entire finishing distribution matrix in $O(n \times 2^n)$ by utilizing a 2^n sized cache for memoization.

Instead of enumerating all elimination sequences, we enumerate the subsets of players still in the tournament. While still exponential, it provides a considerable improvement over the factorial runtime of the naive implementation.

In Malmuth-Harville players are removed from the remaining sub-calculation when they finish in a certain spot. This means that the sub-calculation carried out following the finish of players i and j in the first two places does not depend on the finishing order between i and j. These sub-calculations can therefore be carried out in a combined step, instead of individually.

In our improved implementation, we calculate the weight W for each subset of players $N' \subseteq N$ finishing in the first $|N'|$ places. We start by calculating the weight of each subset with $|N'| = 1$ using the original assumption 2.1. For subsets with $|N'| = 2$, we can then efficiently calculate the weights by re-using the $|N'| = 1$ results and apply 2.2. As we incrementally increase the size of the calculated subsets, the weight of each subset can be calculated using the weights calculated during the previous iterations. Each step uses the weights of the $|N'|$ subsets that can lead to the current N'. For the calculation of all 2^n subsets, this leaves us with a total runtime complexity of $O(n \times 2^n)$.

The weight cache W can be efficiently implemented as a simple array, indexed by a n-bit integer representing the 2^n subsets. An example implementation in Java is included in listing A.1.

Algorithm 2.1: Optimized Malmuth-Harville "ICM"

```
Data: Chip stacks $x_1 \dots x_n$, Set of players $N = \{1 \dots n\}$
Result: Finishing distribution matrix P
1  Initialize $P = \mathbb{R}^{n \times n}$ with $p_{i,j} = 0$
2  Initialize $W[N' \subseteq N] = \begin{cases} 1 & \text{if } N' = \emptyset \\ 0 & \text{else} \end{cases}$
3  for $f = 1 \dots n$ do
4      foreach $N' \subseteq N$ with $|N'| = f$ do
5          $r = \sum_{i \notin N'} x_i$
6          foreach $i \in N'$ do
7              $t = \frac{x_i}{r + x_i} W[N' \setminus i]$
8              $p_{i,f} \leftarrow p_{i,f} + t$
9              $W[N'] \leftarrow W[N'] + t$
10         end
11     end
12 end
13 return $P$
```

2.2 Malmuth-Weitzman

Malmuth-Weitzman is a less popular variant of the ICM model discussed before. Using the results from [CA06, p.337-339], the model can be stated as a similar set of assumptions:

1. The probability of a player being eliminated from the tournament next is inversely proportional to the player's chip stack.
2. The chips of an eliminated player are distributed evenly among the remaining players.

This is conceptually a "bottom up" version of Malmuth-Harville. Instead of starting by assigning the probabilities of finishing first and enumerating the sequences top-down, Malmuth-Weitzman starts with the player to be eliminated next and works by enumerating the sequences bottom-up to the first place. It is notable that by applying these assumptions, 2.1 also holds true for Malmuth-Weitzman.

2.2.1 Example

We use P_W to indicate the finishing distribution matrix as estimated by the Malmuth-Weitzman heuristic and υ_W for the estimated payoffs. Both are displayed below using the previous example of chip stacks $x = [2, 5, 8, 13, 17]$ and a payout structure of $R = [50, 30, 20, 0, 0]$.

$$P_W(x) = \begin{bmatrix} 0.044 & 0.071 & 0.124 & 0.240 & 0.520 \\ 0.111 & 0.159 & 0.228 & 0.294 & 0.208 \\ 0.178 & 0.223 & 0.257 & 0.212 & 0.130 \\ 0.289 & 0.277 & 0.212 & 0.142 & 0.080 \\ 0.378 & 0.269 & 0.179 & 0.112 & 0.061 \end{bmatrix}$$

$$\upsilon_W(x, R) = P_W(x) \times R^T = \begin{bmatrix} 6.84 & 14.89 & 20.73 & 26.99 & 30.56 \end{bmatrix}$$

2.2.2 Improved Implementation

As with Malmuth-Harville, a naive implementation will enumerate all elimination sequences and the runtime for calculating the finishing distribution matrix therefore grows with $O(n!)$.

With minor modifications, an optimized version similar to 2.1 can be used to reduce the runtime to $O(n \times 2^n)$. This again utilizes a cache of size 2^n and, as in Malmuth-Harville, relies on the property that the stacks used for the sub-calculation after the elimination of a set of players does not depend on the order of their elimination. An example implementation in Java is also included in listing A.2.

Algorithm 2.2: Optimized Malmuth-Weitzman

```
Data: Chip stacks $x_1 \dots x_n$, Set of players $N = \{1 \dots n\}$
Result: Finishing distribution matrix P
1  Initialize $P = \mathbb{R}^{n \times n}$ with $p_{i,j} = 0$
2  Initialize $W[N' \subseteq N] = \begin{cases} 1 & \text{if } N' = N \\ 0 & \text{else} \end{cases}$
3  for $r = n \dots 1$ do
4      foreach $N' \subseteq N$ with $|N'| = r$ do
5          $d = \frac{1}{r} \sum_{i \notin N'} x_i$
6          $b = \sum_{i \in N'} \frac{1}{x_i + d}$
7          foreach $i \in N'$ do
8              $t = \frac{1}{b(x_i + b)} W[N']$
9              $p_{i,r} \leftarrow p_{i,r} + t$
10             $W[N' \setminus i] \leftarrow W[N' \setminus i] + t$
11         end
12     end
13 end
14 return $P$
```

2.3 Roberts

In [Rob11], Roberts introduces a new equity model that is claimed to provide more accurate approximations. It is conceptually similar to Malmuth-Weitzman, but using a slightly different set of assumptions:

1. The probability of a player being eliminated from the tournament next is assumed to be:
$$Pr(X_{i,n}) \propto \frac{1}{x_i^2} \sum_{j \neq i} \frac{1}{x_j} \tag{2.3}$$

2. The chips of an eliminated player are distributed inversely proportional to the chip stacks of the remaining players.

As with the previous two algorithms, a naive implementation of the Roberts heuristic also has a runtime complexity of $O(n!)$. Our optimization for the previous two algorithms is not directly applicable in this case because of the way the chips are redistributed when players are eliminated. When redistributing chips inversely proportional to the chip stacks, the new stacks no longer depend solely on the set of removed players, but also on the order of their elimination.

Fortunately, the same paper [Rob11] also provided an adapted version of this algorithm with a considerably better runtime complexity of only $O(n^4)$ for the calculation of the finishing distribution matrix.[1] This adapted algorithm is an approximation and no longer produces a finishing matrix with column and row sums of exactly 1.

We evaluated both the approximation and the full variant of the algorithm and found that both variants perform essentially the same. For this reason, we have only included the results for the full (non approximated) version in later chapters.

2.3.1 Example

The finishing distribution matrix P_R and estimated payoffs v_R for the Roberts heuristic are shown below, using the same example of chip stacks $x = [2, 5, 8, 13, 17]$ and $R = [50, 30, 20, 0, 0]$.

$$P_R(x) = \begin{bmatrix} 0.044 & 0.049 & 0.073 & 0.144 & 0.690 \\ 0.111 & 0.124 & 0.191 & 0.392 & 0.182 \\ 0.178 & 0.197 & 0.290 & 0.256 & 0.078 \\ 0.289 & 0.301 & 0.252 & 0.127 & 0.031 \\ 0.378 & 0.329 & 0.194 & 0.081 & 0.019 \end{bmatrix}$$

$$v_R(x, R) = P_R(x) \times R^T = \begin{bmatrix} 5.16 & 13.09 & 20.61 & 28.51 & 32.63 \end{bmatrix}$$

2.4 Model Limitations

The equity models discussed in this chapter are taking only the absolute chip stacks and payout structure as input variables. Besides these factors, actual payoffs in tournament states also depend on the absolute and relative position of the chip stacks, as well as the relative size of the chip stacks compared to the blinds.

A common example of this shortcoming is that single hand solutions based, on the models discussed previously, will typically over-estimate the expected payoff for players in the positions left of the blinds in the case of a fold; this results in too passive play.

The reason, is that, with few exceptions, the SB and BB positions are expected to lose chips. When approximating payoffs only based on the chip stacks, without considering

[1]The paper states a runtime of $O(n^3)$, but this refers to the equity calculation for a single player, not the entire matrix.

positions, this necessarily fails to take into account the loss of chips during the next hand for the players about to enter the blind positions. As larger chip stacks tend to lose a smaller fraction of their value in the blinds, there is an additional incentive for players to increase their chip stacks before entering the blind positions.

Leaving aside positional issues, the Malmuth-Harville model is often claimed to underestimate expected payoffs for large chip stacks [Rob11]. We will see in chapter 7 that this is indeed the case in our evaluation.

CHAPTER 3

Single Hand Equilibrium

3.1 Nash Equilibrium

Nash equilibria are one of the standard solution concepts for multiplayer games. A set of strategies is called a Nash equilibrium if, given perfect knowledge of the other players' strategies and treating them as static, none of the players has an incentive to unilaterally change their strategy. In other words, a Nash Equilibrium is a set of strategies where each of the strategies is a best response against the other strategies.

For two player constant sum games, a Nash equilibrium is also the minimax solution and comes with strong guarantees associated. For multiplayer games there are generally no such strong guarantees.

Exact Nash equilibria are notoriously hard to find, especially so for big stochastic games. Instead the concept of ϵ-Nash equilibria is widely used as an approximation.

A set of strategies is called an ϵ-Nash equilibrium iff no player can increase his expected payoff by more than a fixed value ϵ when unilaterally deviating from their assigned strategy.

3.2 Iterative ϵ-Nash Approximations

In the following sections we will describe some popular algorithms for approximating ϵ-Nash equilibria. The discussed algorithms are all based on simulated self-play. In all cases, the agents start out with an arbitrary strategy and then simulate play; making incremental adjustments to their strategy profiles.

All three algorithms discussed in this section have memory requirements linear to the number of information sets (not game states). The algorithms traverse over the information sets during each iteration and perform somewhat similar update procedures.

3.3 Fictitious Play

Fictitious Play[Bro51] [FL98] is a popular learning model used to iteratively calculate an ϵ-Nash Equilibrium. It uses a sequence of simulated games. In each iteration, each player i uses a strategy profile that is a best response against the average of the previously observed strategy profiles used by the other players.

The next strategy profile for player i is therefore:

$$\sigma_i^{T+1} = b_i(\bar{\sigma}_{-i}^T) \tag{3.1}$$

Where:

- $\bar{\sigma}_{-i}^T$ is the average strategy profile for the other players for iterations $1 \dots T$.
- $b_i(\bar{\sigma}_{-i}^T)$ selects a best response for player i against $\bar{\sigma}_{-i}^T$.

The Fictitious Play algorithm does not specify how the best response is selected, in case multiple actions have the same utility. As explained below, it is generally beneficial for performance to keep the number of actions with mixed strategies low, so that the best response may be chosen with this in mind. Generally the specific selection function is not particularly important in poker games, because it is a rather rare occurrence to have multiple actions with exactly the same utility.

In its standard form, one of the problems with using Fictitious Play in poker is that it results in profiles with a high number of mixed strategies. With Fictitious Play, if an action was included in any of the σ^T profiles, the action will remain in the average strategy profile $\bar{\sigma}$ with a non-zero probability throughout the remaining calculation. This leads to a large number of “almost pure” action profiles. We can see an example of this in the strategy tables included in [GS08], which include a considerable number of entries at 99% or 1%.

The runtime for evaluating a terminal node typically depends on the number of information sets for each player that reach the node with non zero probability. The exact runtime complexity is implementation dependent, and can often be considerably improved by exploiting domain specific properties of the game structure. Some options for improving the evaluation of terminal nodes in poker games are discussed in [JWBZ11].

Pure strategies are often beneficial to the evaluation performance, because whenever an action is played with a pure strategy the information set will have a probability of zero in some of the remaining game branches. For this reason, the tendency of Fictitious Play to include a large number of mixed (“almost pure”) strategies typically has a negative impact on evaluation speed.

3.4 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) [Joh07][ZJBP08] is an iterative regret minimization algorithm. The algorithm is somewhat similar to Fictitious Play in terms of implementation. Both algorithms use the average strategy profile $\bar{\sigma}$ played during the iterations to find an ϵ-Nash equilibrium.

Beside storing the average strategy profile, CFR also maintains the cumulative counterfactual regret values $R_i^T(I,a)$ for every action a of player i in every information set I, and uses these cumulative regrets to construct the current strategy profile for each player. The counterfactual value $\upsilon_i(\sigma, I)$ for strategy profile σ in information set I is defined as:

$$\upsilon_i(\sigma, I) = \pi_{-i}^{\sigma} u_i(\sigma, I) \tag{3.2}$$

Where:

- π_{-i}^{σ} is the probability of reaching information set I when all players use strategy profile σ, except that player i tries to reach information set I.
- The counterfactual utility $u_i(\sigma, I)$ is the expected utility for player i given that information set I is reached and all players use the strategy profile σ, except that player i plays to reach I.[Joh07] [1]

The cumulative counterfactual regrets are then maintained by the following rule:

$$R_i^T(I,a) = \begin{cases} \upsilon_i(\sigma^t_{I\to a}, I) - \upsilon_i(\sigma^t, I) & T = 1 \\ 0, R_i^{T-1}(I,a) + \upsilon_i(\sigma^t_{I\to a}, I) - \upsilon_i(\sigma^t, I) & T > 1 \end{cases} \tag{3.3}$$

Where $\sigma_{I\to a}$ is a strategy profile identical to σ, except that player i plays action a in information set I.

The strategy profile for the next iteration is constructed by setting the action probabilities proportional to the positive counterfactual regret $R_i^{+,T}(I,a) = max\left(R_i^T(I,a), 0\right)$ if the sum of $R_i^{+,T}$ is positive in that information set, and to a default setting otherwise:

$$\sigma^{T+1}(I)(a) = \begin{cases} \propto R_i^{+,T}(I,a) & \text{if } \sum_{a\in A(I)} R_i^{+,T}(I,a) > 0 \\ \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases} \tag{3.4}$$

As in Fictitious Play, the final results of CFR have a tendency to include a large number of "almost pure" strategies as in both cases the average strategy profile $\bar{\sigma}^T$ is used. However, in the case of CFR this does not have the same negative impact on runtime

[1] A more formal definition is provided in [JBL+12].

because the averaged strategies are not used during the calculation process. The strategy profiles used during the calculation are derived from the regret values and do not have this tendency.

The original version of the algorithm, also referred to as “Vanilla” CFR, processes the entire game tree in every iteration. In [JBL+12], Johanson et al. introduce several Monte Carlo sampled variants of the algorithm that update only a fraction of the tree in every iteration.

A notable disadvantage of CFR is the higher memory requirement. It generally requires twice the memory compared to Fictitious Play, because the cumulative regrets need to be stored in addition to the averaged strategy profiles.

3.5 CFR$^+$

CFR$^+$ uses a slightly different update rule compared to CFR and, instead, maintains the cumulative counterfactual regret$^+$ $R_i^{+,T}(I,a)$. The only difference to CFR is that the values use a lower cap of zero. The modified update rules as provided in [Tam14]:

$$R_i^{+,T}(I,a) = \begin{cases} max\left\{0, \upsilon_i(\sigma^t_{I\to a}, I) - \upsilon_i(\sigma^t, I)\right\} & T = 1 \\ max\left\{0, R_i^{+,T-1}(I,a) + \upsilon_i(\sigma^t_{I\to a}, I) - \upsilon_i(\sigma^t, I)\right\} & T > 1 \end{cases} \tag{3.5}$$

The construction of the strategy profile for the next iteration σ^{T+1} then uses these $R_i^{+,T}(I,a)$ values in rule 3.4.

The authors observed that, when using this modified rule, the current strategy profile σ^T will itself converge to an ϵ-Nash equilibrium, so storing the averaged strategy profile may no longer be required.

This results in one of the most important improvements of the algorithm: the reduced memory requirements compared to CFR. There is the immediate improvement in memory requirements from not storing the averaged profiles, which brings the basic requirements down to the level of Fictitious Play. The new updating rule also results in a considerable percentage of the stored regret values being exactly zero, and compression techniques can be used to further decrease the memory requirements.

For complex games, CFR$^+$ was observed to converge in a magnitude fewer iterations overall. The authors suspect that this is due to the absence of “regret buildup”. In CFR negative regret can be accumulated and even once an action becomes part of the best response again, it can take several iterations before the action re-enters the current strategy profile because the cumulated negative regret first needs to be compensated before a positive value is reached. In CFR$^+$ the regret values use a floor of zero, which results in a more dynamic sequence of strategy profiles.

CHAPTER 4

Tournament Equilibrium

In chapter 2, we discussed models to approximate the expected payoffs for players based on their chip stacks. In order to evaluate the accuracy of these models, we will now explore methods to calculate the actual payoffs for some tournament games.

In the simplest case a tournament state is defined by the vector of chip stacks, but it can also include additional tournament state information. In real tournament games, the blinds are usually escalated by a predetermined schedule. In such cases the tournament state would also need to include the current blind level, and time or number of hands until the next blind increase. We will assume a constant blind level for the remainder of this work and identify tournament states by the vector of chip stacks.

The two main algorithms, VI-FP and PI-FP, are extensions of the Fictitious Play algorithm discussed in chapter 3. They use Fictitious Play as part of an *inner loop*, to calculate equilibrium approximations individually for each tournament state while assuming fixed payoffs for the other states. In the global *outer loop*, the state payoffs are then updated to reflect the new strategies.

An important restriction of both algorithms is that they need to calculate all tournament states, so game abstractions need to be used to restrict the total number of states. Both the complexity and the abstraction requirements make the algorithms somewhat impractical to apply in end-user software.

The Future Game Simulation (FGS) algorithm is an adversarial search with close similarities to the VI-FP algorithm. Unlike the other two algorithms, FGS is not actually suitable for calculating a full tournament solution, but it can provide single-hand calculations of improved accuracy compared to the use of plain equity models discussed in chapter 2. FGS generally does not require the calculation of all tournament states, which makes it more practical for end-users.

In their standard forms, all algorithms in this chapter apply Fictitious Play from section 3.3 to calculate ϵ-Nash equilibria for individual hands. This only reflects the fact that Fictitious Play is more established than the newer regret minimizing alternatives; Fictitious Play can be substituted by any of the ϵ-Nash algorithms from chapter 3.

4.1 VI-FP

In [GS08], Ganzfried/Sandholm introduce an algorithm to calculate approximate Nash equilibria for poker tournaments. It uses a modified value iteration algorithm where Fictitious Play is used in an inner loop to calculate the single hand equilibria.

In the inner loop beginning at line 5 in algorithm 4.1, VI-FP uses Fictitious Play (or an alternative different algorithm from chapter 3) to calculate an ϵ-Nash equilibrium for every tournament state, assuming static payoffs V^i for the other states.

Once the inner loop is complete, the static payoffs are updated at line 9 by calculating the expected payoff of each tournament state, under the assumption that players use the ϵ-Nash strategies of the inner loop and still using the previous payoffs V^{i-1} as the value for all other states. This process is repeated until the maximum deviation between V^i and V^{i-1} is lower than the tournament goal δ.

Even if VI-FP converges, it is not generally guaranteed to be in a Nash equilibrium. An example is provided in [GS09] where the V^0 payoffs are initialized optimistically for all players, which leads to single hand strategies in the first inner loop that cause an infinitely repeated game.

We would like to point out that, when initializing with any of the equity models from chapter 2, the sum of the initial payoffs in a state with n players will be equal to the sum of rewards R for these places $\sum_{i=1}^{n} v_i = \sum_{i=1}^{n} r_i$. The payoff sum is therefore correct in the initialization, and this property is preserved during the outer loop updates of VI-FP. When using a sensible initialization method, it is not possible to arrive at optimistic payoff estimates for all players in any tournament state. It is not clear whether or not the algorithm may converge in non-equilibrium states at any rate.

The same paper [GS09] also introduces an *ex post* equilibrium check to evaluate whether a strategy profile constitutes an ϵ-Nash equilibrium. This check can be used to verify the quality of VI-FP results after the strategy calculation.

Algorithm 4.1: VI-FP [GS08]

```
Data: Fictitious Play goal $\gamma$, Tournament goal $\delta$
Result: Tournament strategy $\sigma$
1  $V^0 = initializePayoffValues()$;
2  $i = 0$;
3  repeat
4      $\sigma \leftarrow initializeStrategies()$;
5      repeat
6          $\sigma \leftarrow fictPlay(V^i)$;
7      until $maxRegret(\sigma) \leq \gamma$;
8      $i \leftarrow i + 1$;
9      $V^i \leftarrow getNewValues(V^{i-1}, \sigma)$;
10 until $maxDev(V^i, V^{i-1}) \leq \delta$;
11 return $\sigma$;
```

4.2 Future Game Simulation

FGS is a popular extension to improve the equity estimates of the equity models discussed in chapter 2. It is an adversarial search; essentially an adaptation of the expectiminimax [Mic66, p.183-200] algorithm for poker tournaments.

The general concept is similar to the minimax search used for games such as chess, where a number of future moves are calculated and, at the deepest level, a heuristic evaluation is used to estimate the expectation in the remainder of the game.

Similarly, in FGS the tournament states reachable within a specified number of hands are calculated, and at the deepest level, one of the equity models mentioned before, typically Malmuth-Harville, is used to approximate the expected payoff for the remainder of the tournament.

The main difference between FGS and other adversarial searches, such as expectiminimax, is that each depth level in the FGS calculation represents an entire hand of poker with decisions by multiple players. Because of the imperfect information during hands, expectiminimax can not be used directly to calculate the player actions. Instead, FGS calculates an ϵ-Nash equilibrium for each hand and uses the resulting strategy profile to determine the transition probabilities to the next level.

It is notable that a plain implementation of FGS is actually equivalent to the VI-FP algorithm 4.1, when Malmuth-Harville is used as initialization and a fixed count of outer iterations is used instead of VI-FP's δ goal. This outer iteration count corresponds to the depth parameter d in FGS.

The practical difference between the methods is that VI-FP requires a game abstraction that limits the total number of tournament states. This is usually achieved by selecting a suitable starting state where all chip stacks are multiples of a SB, along with a modification of the showdown rules regarding *split pots* to ensure that chip stacks remain multiples of a SB throughout the entire abstract tournament. An example for a suitable game abstraction is discussed in section 5.3.5.

FGS only calculates the tournament states that are reachable from an arbitrary starting state within the search limit specified by the depth parameter d, so it is not necessary to restrict the overall number of tournament states. However, the number of reachable tournament states without tiebreakers will typically increase exponentially with the depth limit and, therefore, only relatively small limits are used in practice, most commonly in the range of 1-5 rounds.

Algorithm 4.2: Future Game Simulation "FGS"

```
Data: Chip stacks $x$, Fictitious Play goal $\gamma$, Depth $d$
Result: Payoff estimates $v^d(x)$
1  if $d = 0$ then
2      $v^0(x) \leftarrow MalmuthHarville(x)$;
3      return $v^0(x)$;
4  else
5      forall the $x'$ successor of $x$ do
6          $v^{d-1}(x') \leftarrow FGS(x', \gamma, d-1)$;
7      end
8      $\sigma(x)^{d-1} \leftarrow initializeStrategies()$;
9      repeat
10         $\sigma(x)^{d-1} \leftarrow fictPlay(v^{d-1}, \sigma(x)^{d-1})$;
11     until $maxRegret(v^{d-1}, \sigma(x)^{d-1}) \leq \gamma$;
12     $v^d(x) \leftarrow getNewValue(x, v^{d-1}, \sigma(x)^{d-1})$
13     return $v^d(x)$;
14 end
```

4.2.1 Common Variants

The transition probability from $x \to x'$ is typically very low if the state x' can only be reached from x as a result of split pots. If no tiebreaker rule is used, these low-probability branches may be terminated by FGS implementations before the full depth limit is reached and their payoffs replaced by the heuristic equity model estimates. This optimization alone results in a reduction of the branching factor of approximately 50% in popular game abstractions, with the exact figure depending on the player count.

A second common simplification is the use of a restricted strategy space for the Fictitious Play calculations carried out as part of FGS. This option is discussed in more detail in section 5.4.

4.3 PI-FP

The PI-FP [GS09] algorithm is a follow up to the VI-FP version, based on the policy iteration algorithm [Put05]. Compared to VI-FP it uses a modified outer loop to update the payoffs.

In the outer loop update of PI-FP, the transition matrix based on the current strategies is built and the resulting linear system is solved to find the updated state payoffs. If there are multiple solutions, the minimal non-negative solution is selected.

The updated payoffs in the outer loop are only dependent on the current strategy profile, not on previously assigned payoff values. This means that, unlike VI-FP, the PI-FP variant can recover from poor initialization and is guaranteed to be in a Nash equilibrium if the algorithm converges.

Payoffs may still be initialized by one of the equity models from chapter 2 for the first inner loop, but compared to VI-FP initialization, is much less important.

Neither VI-FP nor PI-FP define how the strategies are initialized as part of the *initializeStrategies()* step. In practice it is advisable to initialize with the last iteration's strategies. After a few outer iterations, the payoff updates are usually relatively small and initializing with the previous strategies can drastically reduce the runtime compared to a random initialization.

With the relatively recent publication of CFR$^+$, there is now a promising ϵ-Nash algorithm available that does not require the use of an averaged strategy. As discussed in section 3.5, this results in a more dynamic series of strategy profiles, and if used as an inner-loop replacement of Fictitious Play we suspect that CFR$^+$ can cope with the payoff updates of the outer loop without a reset of the regret values.[1]

[1] As a result of their averaged nature, the solutions of Fictitious Play and CFR can not change rapidly in later iterations, so they would likely require an adjustment or reset to work with the outer loop payoff updates.

We will use a variant of PI-VP based on this idea to calculate the tournament solution, the algorithm being outlined in chapter 6.

Algorithm 4.3: PI-FP [GS09]

```
Data: Chip stacks $x$, Fictitious Play goal $\gamma$, Tournament goal $\delta$
Result: Tournament strategy $\sigma$
1  $V^0 = initializePayoffValues()$;
2  $i = 0$;
3  repeat
4      $\sigma = initializeStrategies()$;
5      repeat
6          $\sigma = fictPlay(V^i)$;
7      until $maxRegret(\sigma) \leq \gamma$;
8      $i = i + 1$;
9      $M^i = createTransitionMatrix(\sigma)$;
10     $V^i = evaluatePolicy(M^i)$;
11 until $maxDev(V^i, V^{i-1}) \leq \delta$;
12 return $S^i$;
```

CHAPTER 5

Game Abstraction

5.1 Jam/Fold Play

The two-player Limit Hold'em poker variant was recently "weakly solved" by Johannson et al in early 2015, using only lossless abstractions.[BBJT15] Even after exploiting symmetries in the game, two-player Fixed Limit Hold'em has 1.38×10^{13} information sets. Uncompressed, this game requires 262TB of storage for storing the solution, although this can be reduced to 11TB after utilizing compression techniques.

The No Limit Hold'em (NLHE) variant of poker is several orders of magnitude more complex. Even for a two player game, solving a NLHE game without abstractions is considered solidly out of reach for the foreseeable future. The current work on NLHE uses restrictive lossy abstractions to deal with the complexity.

For tournament games, a popular abstraction is to limit the set of actions available to the players to either jam (go *all-in*) or fold. Fortunately, tournament settings often deal with a relatively low ratio of chip stacks to blinds so this is not likely a large mistake.

In Mathematics of Poker [CA06], Chen/Akenman argue that jam/fold play is likely optimal for stack sizes up to roughly 7×BB and a reasonable approximation up to around 10×BB. It is, however, worth noting that their argument is based on chip values and ignores tournament considerations. For human play, a wide variety of tournament learning material generally recommends the use of jam/fold strategies up to chip stacks of approximately 10×BB.

5.2 Ganzfried/Sandholm Three Player Game

We will first look into the abstract game that Ganzfried/Sandholm used in [GS08] and [GS09] and then determine additional simplifications to allow more than 3 players.

Their game was based on the end game of 9-player STTs on PokerStars[1]. These games start with 9 players and 1500 chips per player and use a payout structure of 50% for 1st, 30% for 2nd and 20% for 3rd place. For the sake of simplicity they assumed a total prize pool of 100$.

In the original PokerStars game format, the blind levels are escalated by a fixed time schedule. In the abstract game, a fixed blind level of SB=300 and BB=600 was used. Additionally, the game used a tiebreaker rule so split pots would be avoided. The significance of the tiebreaker rule is that it drastically reduces the possible tournament states when using jam/fold play, and has a starting state where all players have stacks that are full multiples of a SB. This ensures that player stacks remain multiples of a SB for the entire tournament and effectively leaves us with $13500/300 = 45$ stack units that cannot be broken down in the abstract game.

As shown in table 5.3, this results in a total of 946 different tournament states for 3 players and 45 stack units. It is well known that winning probabilities for the two player game are approximately proportional to the players' chip stacks. For this reason, [GS08] and [GS09] only calculated the three player tournament states and used the approximation above to estimate the payoffs for the two player states.

5.3 Complexity Considerations

One essential implementation detail that enables us to efficiently calculate three player hands is the use of a pre-calculated lookup table that provides the result probabilities for all possible hand matchups. The technique is described in some detail in [GS08].

When two or more players choose to go all-in, the community cards are dealt and there is a showdown to determine which player wins the pot. For three players there are a total of 6 private hole cards removed from the deck before dealing the board, which leaves us with a total of $46C5 = 1{,}370{,}754$ board run-outs. Even using highly optimized hand evaluators, enumerating this on-the-fly would be several magnitudes too slow in practice.

Instead of enumerating these boards on-the-fly during the calculation, a pre-calculated lookup table is used that stores the outcome probabilities for every possible matchup of hole cards. Since there are no community cards when players make their decisions in the jam/fold game, the card suits are entirely symmetrical and we can reduce the $52C2 = 1{,}326$ hole card combinations into 169 strategically distinct preflop hand classes.

5.3.1 Rollout Size

When adjusting the game rules for additional players, we first need to decide whether it is feasible to use a bigger lookup table to account for all matchups, or if other abstractions should be considered that enable us to keep working with the three player lookup table.

[1]PokerStars is the biggest provider of online poker as of 2015, `http://www.pokerstars.com`

Ignoring split pots, there are $n!$ possible results for every hand matchup. The number of naive matchups is 169^n, but it is not necessary to calculate the full number as many matchups are isomorphic. We can restrict the calculation to cases of $H_1 \geq H_2 \geq \cdots \geq H_n$, where each of the H_i is one of the 169 Preflop classes. The order of hand classes can be defined arbitrarily as it is only used for indexing the lookup table. Table 5.1 provides an overview of the lookup size depending on the number of indexed players.

Players	Results	Matchups	Entries
2	2	14,365	28,730
3	6	818,805	4,912,830
4	24	35,208,615	845,006,760
5	120	1,218,218,079	146,186,169,480

Table 5.1: Size of the rollout lookup depending on players.

When calculating Nash equilibria for single hand scenarios, the vast majority of runtime is spent calculating these outcome frequencies, and it is essential for performance that the lookups be held in RAM. Apart from the memory requirements, the worst case runtime for calculating the combined frequencies also grows exponentially by 169^n with the number of players (in practice, the runtime degrades even more because of the increased percentage of cache misses as the lookup table grows).

Using 8-byte doubles for each of the entries, the 4-player lookup table seems borderline feasible at 6.3GB, but at 1.1TB the 5-player lookup is clearly out of question with our available hardware.

Using a 4-player lookup would generally restrict the calculations to four players due to the additional complexity per hand. The alternative is to use the much smaller 3-player lookup table with 37.5MB, similar to [GS08] and select a game abstraction that avoids the requirement of a 4+ player lookup table. This option appeared more appealing, as it considerably softens the complexity growth per hand when adding players and therefore opens the possibility to calculate the game for more than four players.

5.3.2 Rollout Abstraction

Due to the considerations discussed in the previous section, we now have to introduce additional restrictions on the game when adding more players, so the game can be calculated with a lookup table for no more than three players. The following abstraction is commonly used in commercial poker analysis software.

The adjusted rules for this abstract game are as follows:

1. Once three players are all-in, all remaining players are forced to fold
2. Players receive their hole cards from the deck the first time they act during the hand

3. If players fold, their hole cards are re-shuffled into the deck before action continues

This restricts the maximum number of active players to three, and once that number is reached all remaining players are forced to fold. Shuffling folded hole cards back into the deck is necessary to avoid the requirement of a bigger lookup table, otherwise folded hole cards also need to be taken into account when calculating the outcome frequencies.

It is notable that this abstraction makes our 3-player hands slightly different from the ones in the original 3-player game of [GS08] and [GS09]. In our game abstraction, the subgame after a player folds is independent of the hole cards dealt to the folded player because the cards are shuffled back into the deck before the rest of the hand continues. In the real game, folded hole cards influence the distribution of outcome frequencies.

The effect of folded cards on the remainder of the hand is referred to as “card bunching” in the poker community. It is assumed to be negligible when the number of folded players is low. For this reason, we still expect our three player results to closely match the ones produced by Ganzfried/Sandholm.

5.3.3 Complexity of Single Hand Calculation

We now take a look at complexity of single hand calculations when using the rules outlined so far. Table 5.2 shows how the number of decision points scales with the number of players in a jam/fold game where no more than three active players are allowed.

Players	**1-Way**	**2-Way**	**3-Way**	**Total**
2	1	1	0	2
3	2	3	1	6
4	3	6	4	13
5	4	10	10	24
6	5	15	20	40
7	6	21	35	62
8	7	28	56	91
9	8	36	84	128
10	9	45	120	174

Table 5.2: Decision nodes in jam/fold games with ≤ 3 active players.

For hands with more than three players, a large majority of the runtime is spent calculating the 3-way result frequencies, in our implementation typically in excess of $> 95\%$. For this reason, we will use the number of 3-way decision points as a rough approximation of the complexity when estimating the overall tournament complexity in the next section.

5.3.4 Tournament States

The number of tournament states is dependent upon both the number of players and the total number stack units to be distributed among players. The original Ganzfried/Sandholm game used 45 units, each corresponding to 1 SB, and used a tiebreaker rule for split pots to ensure these units are not broken up. Table 5.3 shows the number of tournament states depending on the number of units and players. As a reference we also included the number of states without a tiebreaker rule.

Table 5.3 shows the number of tournament states for a specific player count, not including the subgames with lower player counts. The column with 45 units corresponds to the original Ganzfried/Sandholm 3-player game with an SB of 300; the 90 units version is for a SB of 150. For reference, purposes we also included the number of tournament states for $13,500$ total chips when no tiebreaker rule is used.

Players	45 Units	90 Units	No Tiebreaker
2	4.40×10^{1}	8.90×10^{1}	1.35×10^{4}
3	9.46×10^{2}	3.92×10^{3}	9.11×10^{7}
4	1.32×10^{4}	1.14×10^{5}	4.10×10^{11}
5	1.36×10^{5}	2.44×10^{6}	1.38×10^{15}
6	1.09×10^{6}	4.15×10^{7}	3.73×10^{18}
7	7.06×10^{6}	5.81×10^{8}	8.39×10^{21}
8	3.83×10^{7}	6.89×10^{9}	1.62×10^{25}
9	1.77×10^{8}	7.06×10^{10}	2.73×10^{28}

Table 5.3: Tournament states for various games.

Table 5.4 shows the total number of tournament states for the 45 unit game when starting with the specified player count, this time including states with fewer players. The approximated complexity is based on the total number of 3-way decision nodes in the tournament. The feasible limit to run the calculations within the allocated time and hardware restricts us to at most 5 players, around $1,500$ times the calculation complexity of the original 3-player tournament game used in [GS08] [GS09]. A further increase to 6 players would result in an additional complexity growth of several magnitudes.[2]

5.3.5 Tiebreaker Rule

By standard game rules, if multiple players have the strongest hand of the exact same rank at showdown, the pot is distributed equally among these players. This can result in the breaking down of SB units into smaller fractions for example, when a pot with an uneven number of SBs is tied by two players.

[2]A game variant with 6 players and 45 stack units is also unappealing because the game has no natural starting state, whereas the 5 player game can start with equal stacks of 9 units per player.

Players	Total States	Complexity
3	990	1
4	14,234	57
5	149,985	1,492
6	1,235,993	24,452
7	8,295,045	285,622
8	46,615,613	2,554,070
9	223,848,240	18,291,428

Table 5.4: Tournament complexity for the 45 unit game.

Using a tiebreaker rule is essential to restrict the number of tournament states to chip configurations where all chip stacks are multiples of a SB. In the event of a tied pot in the abstract game, we break ties between winning hands at random.

5.4 Restricting Strategy Space

A popular abstraction used in tournament analysis software is to further restrict the strategy space of jam/fold games by applying a "hand ranking".

A hand ranking is a fixed order of the 169 Preflop hand classes; the "Sklansky-Chubukov" ranking [Skl99] is a common example. When restricting the strategy space by a hand ranking, players are only allowed to choose a strategy s with $0 \leq s \leq 169$ to play exactly the top s hands of the hand ranking.

This reduces the strategy space of every decision node to 170 pure strategies, in comparison to the 2^{169} pure strategies available in the unrestricted game.

The improvement regarding performance results mainly from the reduced complexity of the terminal node evaluations. Using the unrestricted strategy space, terminal nodes for n players typically have a worst case runtime of $O(169^n)$ because the individual hand matchups from the rollout table need to be combined to get the evaluation for the selected strategies.

In the handranking restricted game, all strategy evaluations can be pre-calculated so the terminal node evaluation has constant runtime $O(1)$ for a single lookup. In the restricted game, the size of this strategy lookup table is identical to the size of the lookup for individual hands we discussed earlier in table 5.1.

It is well known that this abstraction is lossy, regardless of the choice of the hand ranking. As shown for instance in [CA06] and [GS08], there exists no single correct ranking of the Preflop hand classes.

The abstraction is popular for applications where the actual hand selection is not particularly important, for instance, as part of a FGS implementation from chapter 4.2.

In this case, only the payoff estimates are relevant and the actual strategies are usually discarded after the calculation.

We chose to not use this abstraction for the remainder of this work and allow the full strategy space for better accuracy.

5.5 Abstraction Summary

Our tournament setting is essentially a variation of the 3 player jam/fold tournament game used in the Ganzfried/Sandholm papers. We also use 45 SB units to be distributed among players, mimicking the 300/600 blind level with 13,500 total chips, as seen in 9-player PokerStars STTs. A tiebreaker rule is used to handle split pots, which results in a total of 149,985 tournament states.

The total of 45 SBs (22.5 BBs) results in average chip stacks between 4.5 – 11.25 BBs. This is a range where jam/fold is thought to be a reasonable approximation of optimal play[GS08][CA06].

For the reward structure we keep $R = (50\$, 30\$, 20\$)$ for the first three places, making the total prize pool 100$.

To make the terminal node evaluation reasonably efficient, we restrict the number of active players to a maximum of three, and use slightly modified rules regarding card dealing to avoid card bunching effects in our abstract game. These settings result in a game with a total of 5.8×10^8 information sets.

CHAPTER 6

Calculation Overview

6.1 Single Hand Strategies

For the calculation of single hand strategies, we chose to use CFR$^+$ 3.5. A big factor for this decision was that the algorithm appears to work well if payoffs are updated at some point in the calculation. This will be important in our case, especially for the full tournament calculation and for FGS calculations. Instead of entirely re-calculating the hand strategies whenever the payoffs update, we chose to keep the cumulative regrets and simply continue the calculation. This works without any adjustments when using CFR$^+$, when applied to Fictitious Play or “Vanilla" CFR, some reset of the calculation would be necessary as the averaged nature of these algorithms would otherwise prevent fast adjustments of strategies after payoff updates.

We used a target of $\epsilon = 0.0001\$$ (equivalent to 0.0001% of the total prize pool) for all of the single hand equilibrium calculations; both for individual calculations using equity models, and also as the γ goal for the inner loop of the tournament calculation. This is considerably tighter than the $\epsilon = 0.001\$$ limit used in the original 3-player jam/fold paper [GS08].

6.2 Full Tournament Calculation

For the full tournament solution, we essentially used the PI-FP algorithm with some minor modifications. Most importantly, we decided to use CFR$^+$ instead of Fictitious Play for the inner loop calculation, and kept the inner loop regrets stored between outer loop updates. Inner loop updates are only carried out on states that are exploitable by at least γ after the outer loop update. If a state still meets the γ target, no inner loop updates are performed and the strategies of this state remain unchanged. This version of the algorithm converges when no inner loop updates are required after an outer loop update. We will refer to this variant as PI-CFR$^+$ for the rest of this work.

We also chose to calculate the tournament incrementally. We started with states for ≤ 3 players at first, and, once these reached the desired δ, we added the 4 and then 5 player states. As the subgame with ≤ 3 players does not depend on the strategies in the 4 and 5 player states, the payoffs and strategies for these states remain unchanged during the calculation of the 4 and 5 player states. Thus we reduce the total changes in payoffs during the outer loops for the calculation of the much slower 4 and 5 player calculation. We expect this incremental calculation to result in a minor reduction of the overall runtime, compared to a direct calculation of all states.

6.3 FGS

As the game abstraction already limits the number of tournament states, we used VI-CFR$^+$, with a fixed number of outer loops, initialized by Malmuth-Harville, to calculate the FGS results. It is important to note that none of the common FGS optimizations discussed in 4.2 were used. The split pot optimization is obsolete in our game abstraction because of the tiebreaker rule, and further restriction of the strategy space was not necessary due to the already limited number of tournament states.

The PI-CFR$^+$ version discussed above is of course much more efficient for calculating a tournament equilibrium; in fact, it converged to a much closer approximation of the tournament equilibrium in a fraction of the total runtime for 10 rounds of VI-CFR$^+$.

Our interest with FGS is not to fully solve the tournament in this setting, but to evaluate if and by how much a calculation with fixed outer loops improves the equity estimates. As already discussed in section 4.2, unlike the VI/PI algorithms, a fixed-depth FGS calculation can be carried out quite efficiently and on modest hardware from an arbitrary starting state, even with many players and/or the absence of a tiebreaker rule.

CHAPTER 7

Equity Comparison

7.1 Deviation Statistics

This chapter provides a comparison of the equity estimates from various models, compared with the state payoffs from the PI-CFR$^+$ calculation. We frequently use the Mean Absolute Percentage Deviation (MAPD) as an evaluation metric, because we expect relative deviations to be a better estimator of game performance than absolute deviations. Our rationale is that game decisions are made based on differences between expected payoffs, and an estimation error is more likely to impact the strategies if it is a larger percentage of the overall payoff.

Table 7.1 shows a summary of the relative and absolute deviations. Out of the three "plain" equity models, Malmuth-Harville has the lowest overall MAPD at 7.2% with Roberts at 8.2%. We can also see that, while the gap between the two models is much smaller for the Mean Absolute Deviation (MAD), Malmuth-Harville and Roberts are almost identical at 1.04$ and 1.05$ respectively, indicating that the Malmuth-Harville model has lower average deviations for smaller stack sizes and Roberts is more accurate for bigger stacks. The Malmuth-Weitzman heuristic performs considerably worse than the other two models, with a MAPD of 9.87% and a MAD of 1.53$.

Looking at the FGS statistics, it is notable that even at depth 1 there is already a considerable reduction of the MAPD by more than a quarter compared to plain Malmuth-Harville, down to 5.3%. FGS depth 3, which is still practical for most end-user settings, reduces the MAPD by more than half to 3.47%.

7.2 Position Averaged Analysis

The Malmuth-Harville, Malmuth-Weitzman and Roberts models all strictly consider the distribution of chips for their estimates, but not the relative or absolute position of

	Relative Deviation [%]			**Deviation [$]**		
	MAPD	**Min**	**Max**	**MAD**	**Min**	**Max**
Harville	7.20	−53.8	78.8	1.04	−8.59	10.55
Weitzman	9.87	−46.9	84.7	1.53	−7.61	9.88
Roberts	8.19	−59.3	81.6	1.05	−9.69	11.09
FGS-1	5.30	−40.4	66.6	0.83	−7.32	9.62
FGS-2	4.13	−37.2	61.6	0.66	−6.29	7.51
FGS-3	3.47	−33.4	51.6	0.54	−4.52	5.50
FGS-4	2.85	−31.6	38.8	0.42	−4.46	3.95
FGS-5	2.24	−31.0	25.8	0.34	−3.79	4.11
FGS-6	1.78	−29.7	23.9	0.27	−3.15	2.73
FGS-7	1.46	−21.8	18.8	0.22	−2.71	2.21
FGS-8	1.23	−19.8	21.8	0.18	−2.41	2.49
FGS-9	1.00	−21.4	14.7	0.15	−1.85	1.73
FGS-10	0.84	−15.1	12.2	0.13	−1.41	1.23

Table 7.1: Equity statistics summary

the players. To gain an understanding of the part of the deviations accounted for by positioning, we will evaluate the model estimates in a position neutral setting.

We do this by averaging over all positional permutations of a given chip configuration. For instance, given a state $(5, 10, 30)$, we do not directly use the calculated payoffs for this particular state, but instead use the averaged payoffs for the respective stacks over all the permutations, $\{(5, 10, 30), (5, 30, 10), (10, 5, 30), (10, 30, 5), (30, 5, 10), (30, 10, 5)\}$.

Table 7.2 shows the updated table based on these position averaged payoffs. The MAPD is reduced between 2 and 3 percentage points, with positions accounting between 25.2% (Roberts) and 38.9% (Harville) of the original MAPD values in table 7.1. Also notable is the drastic reduction of the minimum and maximum values.

Figure 7.1 shows the expected payoff for a given chip stack in the PI-CFR$^+$ solution, averaged over all tournament states. As mentioned earlier, we fully calculated the two player state instead of estimating the payoffs, by assuming winning probabilities proportional to the chip stacks as in [GS08] [GS09].

As we can see in the figure, the two player payoffs follow the linear approximation very closely as expected. The largest deviation from this approximation is at state $x = \{8, 37\}$, with the payoff for the player in the SB position under-estimated by 0.045$.

Figure 7.2 shows the difference of mean payoffs of the various models against the PI-CFR$^+$ solution for 5 player tournament states. It is notable that, even in the position-neutral setting, there is some obvious bias of the models. Even more surprisingly, all models follow a very similar pattern when plotted in this way. The plot supports that the

	Relative Deviation [%]			Deviation [$]		
	MAPD	**Min**	**Max**	**MAD**	**Min**	**Max**
Harville	4.40	−21.9	13.7	0.68	−2.36	2.24
Weitzman	7.38	−11.0	25.2	1.30	−4.41	2.79
Roberts	5.94	−44.5	15.7	0.71	−3.00	2.69
FGS-1	2.64	−7.5	7.1	0.50	−2.06	1.28
FGS-2	1.98	−4.0	8.2	0.38	−1.74	1.22
FGS-3	1.43	−4.1	6.3	0.27	−1.35	0.91
FGS-4	0.89	−6.2	4.7	0.18	−0.92	1.05
FGS-5	0.58	−3.6	2.5	0.12	−0.63	0.56
FGS-6	0.45	−3.7	2.2	0.09	−0.46	0.44
FGS-7	0.29	−3.6	1.3	0.06	−0.32	0.30
FGS-8	0.25	−3.6	1.2	0.05	−0.33	0.25
FGS-9	0.19	−2.4	0.7	0.03	−0.23	0.17
FGS-10	0.14	−2.1	0.6	0.03	−0.16	0.14

Table 7.2: Equity statistics summary, position averaged

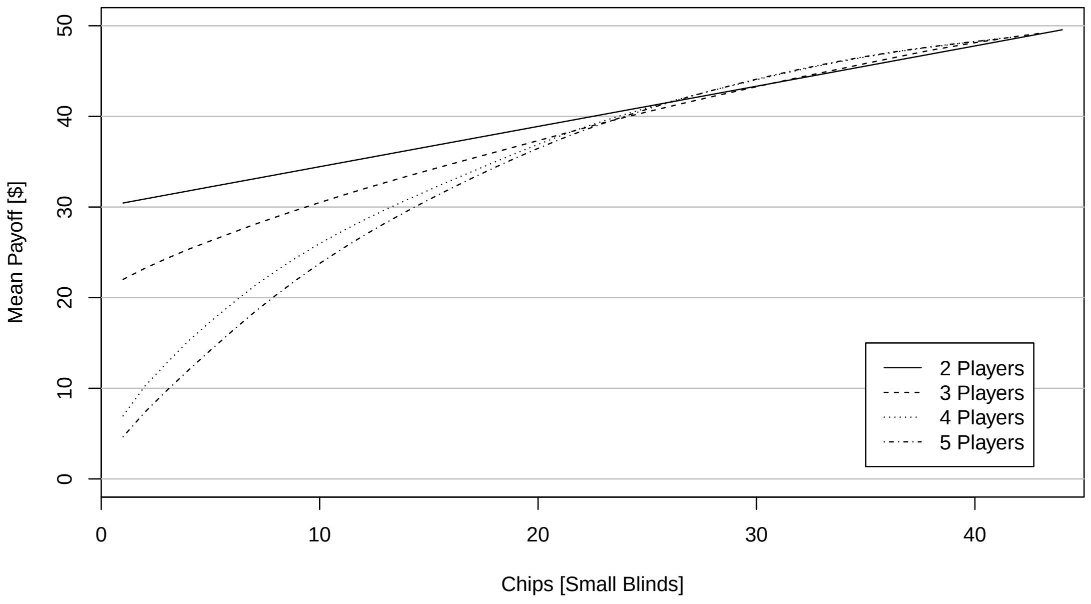

Figure 7.1: Mean payoffs in the PI-CFR$^+$ solution.

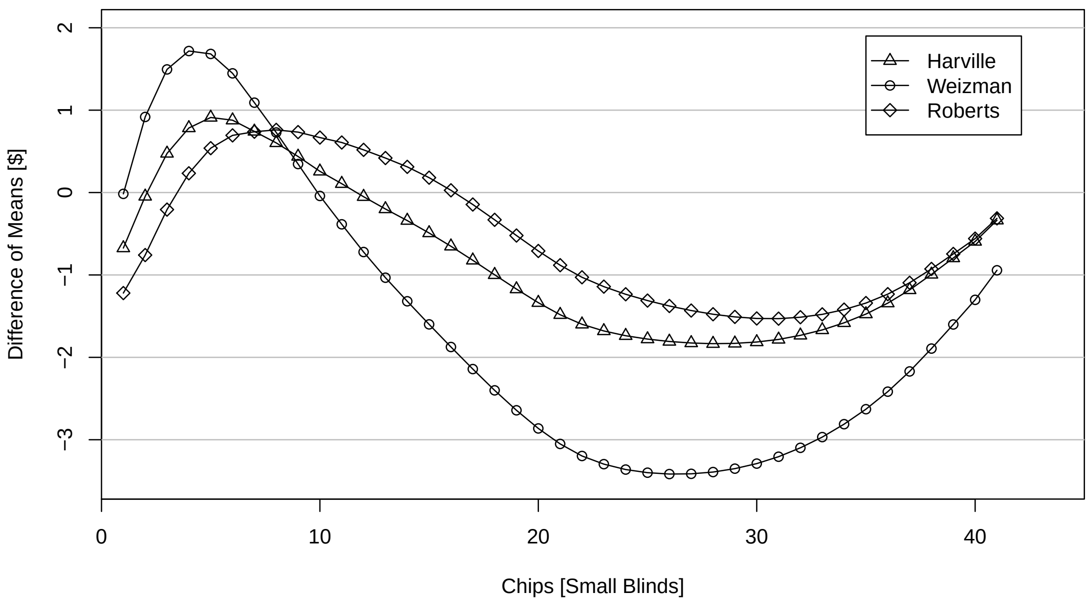

Figure 7.2: Difference of mean payoffs for 5-player states.

Malmuth-Weitzman model provides the poorest estimates of these three models and likely should not be used in practice.

When comparing the Malmuth-Harville and Roberts models, we can see that Roberts has a smaller bias for relatively big stacks, but performs slightly worse than Malmuth-Harville with middle stacks, around 10 to 15 SBs, and considerably worse with very small stacks. Interestingly, all three models systematically under-estimate the payoffs of big chip stacks.

It is notable that this general pattern is also consistent over the three 7.4 and four player 7.3 plots. We omitted the two player plot, as all models estimate the winning changes as proportional to the chip stacks in the two player setting, so their estimates are identical and extremely close to the PI-CFR$^+$ solution.

We observe that the bias pattern applies consistently for three, four and five player tournament states. These are considered to be quite different tournament scenarios. At four players we have the cut-off for the first paid placement, which results in notoriously high non-linearity between expected chips and expected payoffs. (This situation is referred to as “the money bubble” in the poker community.) We can see the quite drastic difference in 7.1 between three and four players, but despite this difference, the general bias pattern remains very consistent.

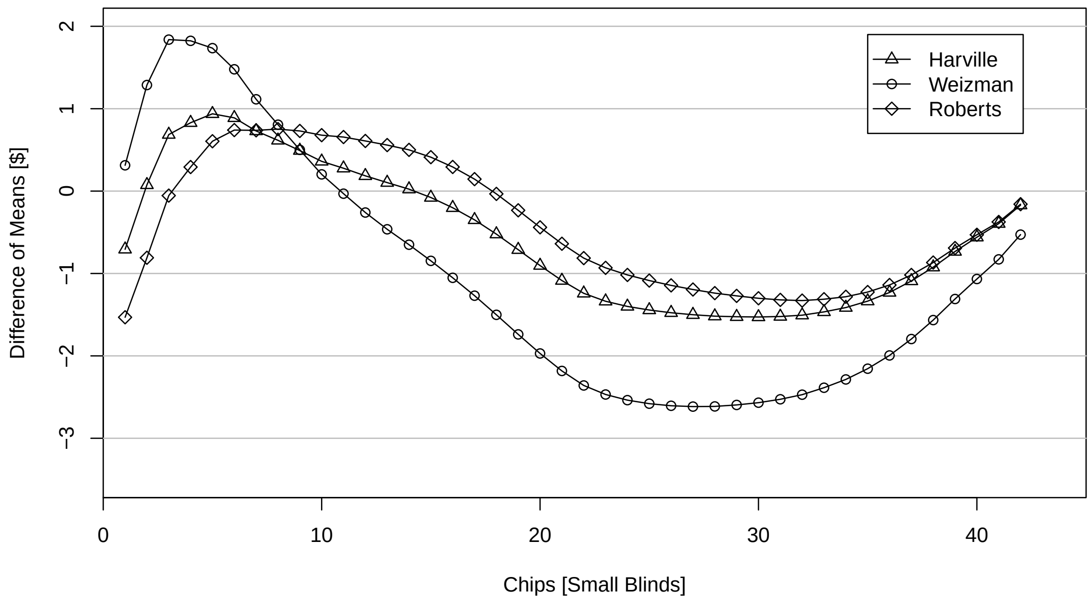

Figure 7.3: Difference of mean payoffs for 4-player states.

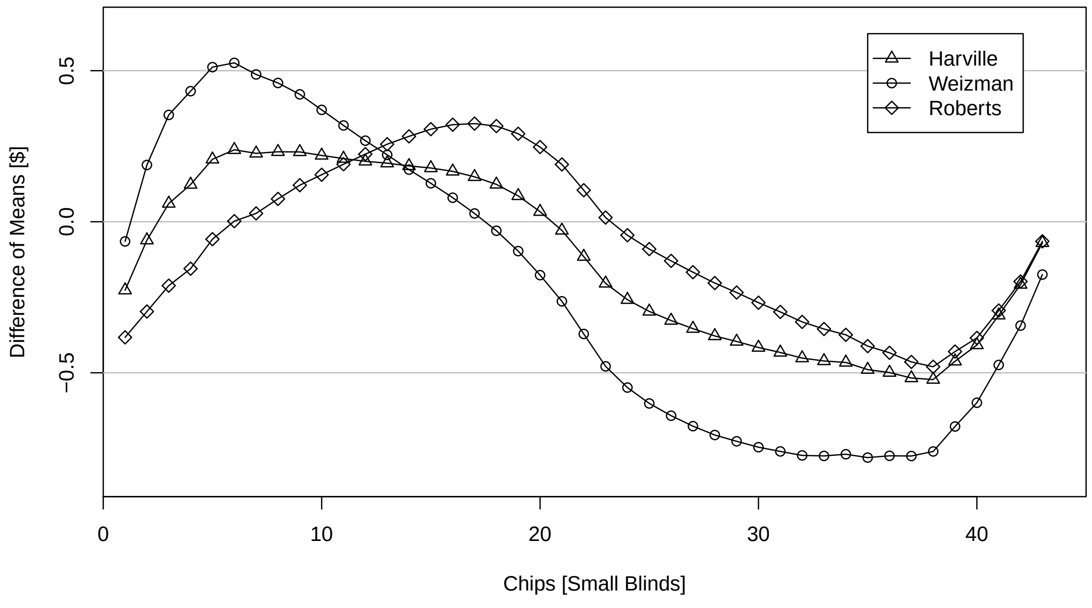

Figure 7.4: Difference of mean payoffs for 3-player states.

7.3 Finishing Probabilities

In this section, we will discuss the average finishing probabilities for players in the PI-CFR$^+$ solution depending on their chip stacks. We can calculate these probabilities by running the payoff update step discussed in the PI-FP section 4.3 on our strategies, using reward structures of $\{(1,0,0,0,0),(0,1,0,0,0),\ldots,(0,0,0,0,1)\}$. The resulting payoffs for these structures are equivalent to the probability of finishing in 1st, 2nd and so on down to 5th place.

The values listed below show the finishing probabilities in the PI-CFR$^+$ solution for the example used throughout chapter 2 with chip stacks of $x = [2,5,8,13,17]$, averaged over the 5! positional permutations of x. We use $P_{\bar{PI}}$ to refer to these position-averaged finishing probabilities in the PI-CFR$^+$ solution.

$$P_{\bar{PI}}(x) = \begin{bmatrix} 0.047 & 0.061 & 0.092 & 0.218 & 0.582 \\ 0.105 & 0.131 & 0.187 & 0.322 & 0.255 \\ 0.166 & 0.204 & 0.273 & 0.245 & 0.112 \\ 0.283 & 0.298 & 0.253 & 0.130 & 0.036 \\ 0.398 & 0.306 & 0.195 & 0.084 & 0.016 \end{bmatrix}$$

$$v_{\bar{PI}}(x,R) = P_{\bar{PI}}(x) \times R^T = \begin{bmatrix} 6.04 & 12.9 & 19.89 & 28.15 & 33.02 \end{bmatrix}$$

All models in chapter 2 assume that the winning probability for players is proportional to their chip stacks. When comparing the winning probabilities in the first column of the matrix above with the calculations for the other models, we observe that the winning probabilities in our solution do not match this assumption. We will discuss this in more detail in the next section.

In tables 7.3, 7.4, 7.5 we provide the average finishing probabilities for our solution, depending on the size of the players' chip stacks in SBs. It is important to keep in mind that these are the average values over all tournament states. The exact probabilities can vary significantly depending on the positions and the distribution of the remaining chips among the other players.

SB	**1.**[%]	**2.**[%]	**3.**[%]	**4.**[%]	**5.**[%]	**SB**	**1.**[%]	**2.**[%]	**3.**[%]	**4.**[%]	**5.**[%]
1	2.50	5.28	11.08	27.99	53.15	26	63.24	28.11	7.39	1.18	0.07
2	4.28	8.28	15.34	28.51	43.59	27	65.73	26.94	6.36	0.92	0.05
3	6.22	11.09	17.72	28.04	36.92	28	68.18	25.69	5.40	0.70	0.04
4	7.99	13.44	20.63	27.43	30.52	29	70.63	24.29	4.54	0.52	0.02
5	9.97	15.82	22.60	27.05	24.56	30	72.95	22.90	3.76	0.38	0.02
6	12.12	18.18	23.99	26.01	19.70	31	75.27	21.38	3.07	0.27	0.01
7	14.24	20.31	25.48	24.17	15.81	32	77.48	19.87	2.45	0.19	0.01
8	16.49	22.36	26.14	22.28	12.73	33	79.60	18.34	1.93	0.13	0.00
9	18.77	24.25	26.48	20.30	10.20	34	81.64	16.80	1.48	0.08	0.00
10	21.04	26.07	26.65	18.22	8.01	35	83.60	15.24	1.11	0.05	0.00
11	23.43	27.61	26.29	16.32	6.35	36	85.40	13.76	0.81	0.03	0.00
12	25.86	29.04	25.70	14.44	4.97	37	87.16	12.24	0.58	0.02	0.00
13	28.37	30.20	24.81	12.72	3.90	38	88.75	10.85	0.39	0.01	0.00
14	31.00	31.12	23.64	11.20	3.05	39	90.01	9.72	0.26	0.00	0.00
15	33.69	31.77	22.32	9.82	2.39	40	91.43	8.41	0.16	0.00	0.00
16	36.42	32.24	20.93	8.55	1.85	41	92.50	7.41	0.08	0.00	0.00
17	39.21	32.47	19.49	7.41	1.42						
18	41.98	32.59	18.04	6.32	1.07						
19	44.77	32.49	16.61	5.34	0.79						
20	47.55	32.27	15.18	4.44	0.57						
21	50.30	31.89	13.77	3.64	0.40						
22	53.01	31.37	12.38	2.95	0.29						
23	55.62	30.76	11.03	2.38	0.21						
24	58.18	30.05	9.73	1.89	0.15						
25	60.73	29.13	8.53	1.51	0.11						

Table 7.3: Average finishing probabilities for 5-player states

SB	**1.**[%]	**2.**[%]	**3.**[%]	**4.**[%]	**SB**	**1.**[%]	**2.**[%]	**3.**[%]	**4.**[%]
1	2.21	7.08	21.75	68.97	26	62.55	29.82	6.94	0.70
2	3.98	11.09	26.61	58.31	27	65.03	28.52	5.91	0.54
3	6.00	14.58	28.22	51.21	28	67.44	27.18	4.99	0.39
4	7.36	17.01	32.75	42.88	29	69.86	25.69	4.16	0.28
5	9.41	19.92	33.35	37.33	30	72.26	24.10	3.44	0.20
6	11.59	22.54	33.57	32.30	31	74.62	22.48	2.76	0.14
7	13.65	24.89	34.12	27.34	32	76.92	20.80	2.19	0.09
8	15.97	26.96	33.54	23.53	33	79.21	19.03	1.70	0.06
9	18.28	28.81	32.68	20.23	34	81.32	17.36	1.29	0.03
10	20.57	30.50	31.77	17.16	35	83.34	15.69	0.95	0.02
11	22.97	31.83	30.39	14.81	36	85.06	14.21	0.72	0.01
12	25.40	32.96	28.94	12.69	37	86.84	12.65	0.51	0.01
13	27.92	33.87	27.30	10.91	38	88.27	11.36	0.36	0.00
14	30.55	34.46	25.56	9.43	39	90.06	9.70	0.24	0.00
15	33.23	34.86	23.74	8.17	40	91.35	8.51	0.14	0.00
16	35.93	35.08	22.02	6.97	41	92.85	7.08	0.07	0.00
17	38.67	35.15	20.25	5.93	42	94.04	5.92	0.03	0.00
18	41.44	35.04	18.57	4.95					
19	44.20	34.93	16.85	4.02					
20	46.96	34.62	15.21	3.20					
21	49.76	34.09	13.62	2.54					
22	52.52	33.38	12.13	1.97					
23	55.11	32.71	10.65	1.53					
24	57.61	31.88	9.32	1.20					
25	60.09	30.93	8.06	0.92					

Table 7.4: Average finishing probabilities for 4-player states

SB	**1.**[%]	**2.**[%]	**3.**[%]	**SB**	**1.**[%]	**2.**[%]	**3.**[%]
1	2.26	14.78	82.96	26	58.70	34.68	6.62
2	4.20	20.94	74.85	27	61.03	33.22	5.75
3	6.31	24.96	68.73	28	63.37	31.84	4.79
4	8.18	29.47	62.35	29	65.71	30.16	4.13
5	10.34	31.77	57.89	30	68.04	28.59	3.37
6	12.49	34.40	53.11	31	70.41	26.73	2.86
7	14.66	36.42	48.91	32	72.73	25.03	2.24
8	16.94	38.06	44.99	33	75.07	23.12	1.81
9	19.24	38.89	41.87	34	77.33	21.28	1.38
10	21.50	40.11	38.39	35	79.74	19.19	1.07
11	23.81	40.43	35.75	36	82.04	17.22	0.74
12	26.10	41.10	32.80	37	84.49	14.99	0.53
13	28.39	41.07	30.54	38	86.71	12.97	0.32
14	30.68	41.36	27.96	39	88.63	11.15	0.22
15	33.02	40.50	26.48	40	90.72	9.17	0.11
16	35.31	40.89	23.81	41	92.55	7.39	0.06
17	37.63	40.34	22.03	42	94.28	5.70	0.02
18	39.95	40.14	19.91	43	95.86	4.14	0.00
19	42.29	39.58	18.12				
20	44.62	39.42	15.96				
21	46.98	38.92	14.10				
22	49.33	38.87	11.80				
23	51.77	37.54	10.68				
24	54.09	36.88	9.03				
25	56.40	35.74	7.86				

Table 7.5: Average finishing probabilities for 3-player states

7.4 Winning Probability

We observed in the previous section that the average winning probabilities in our solution are not proportional to the chip stacks. Figure 7.5 shows the average winning probabilities in the PI-CFR$^+$ solution by player count and chip stack. As we already mentioned earlier, the assumption of winning probabilities proportional to the chip stacks only holds true for tournament states with two players. For states with more than two players, there are some clear deviations.

The foundation for the assumption of proportional winning probabilities is the model of a random walk, where an infinitely small amount of chips is repeatedly re-distributed randomly, and players are removed from the game when left with zero chips[Rob11]. The core idea of this model is that chip movements are entirely random.

In the poker community, it is widely accepted that this is generally not the case for tournament play in situations with high non-linearity between chip-stacks and tournament equity, such as the 4-player payoff "bubble[1]" in our game setting.

Correct strategy for players with small stack sizes is very risk averse in this scenario, as staying in the tournament generally has priority over the accumulation of additional

[1]This refers to the tournament situation before a guaranteed payoff is reached. In our game we have rewards of $R = (50\$, 30\$, 20\$)$ for the first three places, so the player eliminated in 4th place is the last one with a payoff of zero.

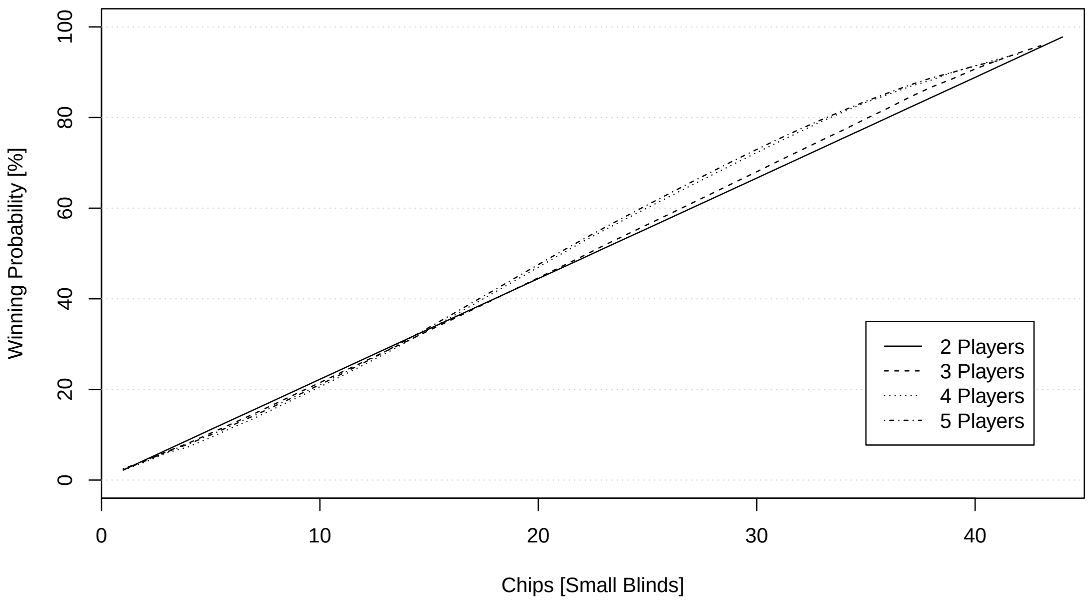

Figure 7.5: Winning probabilities in the PI-CFR$^+$ solution.

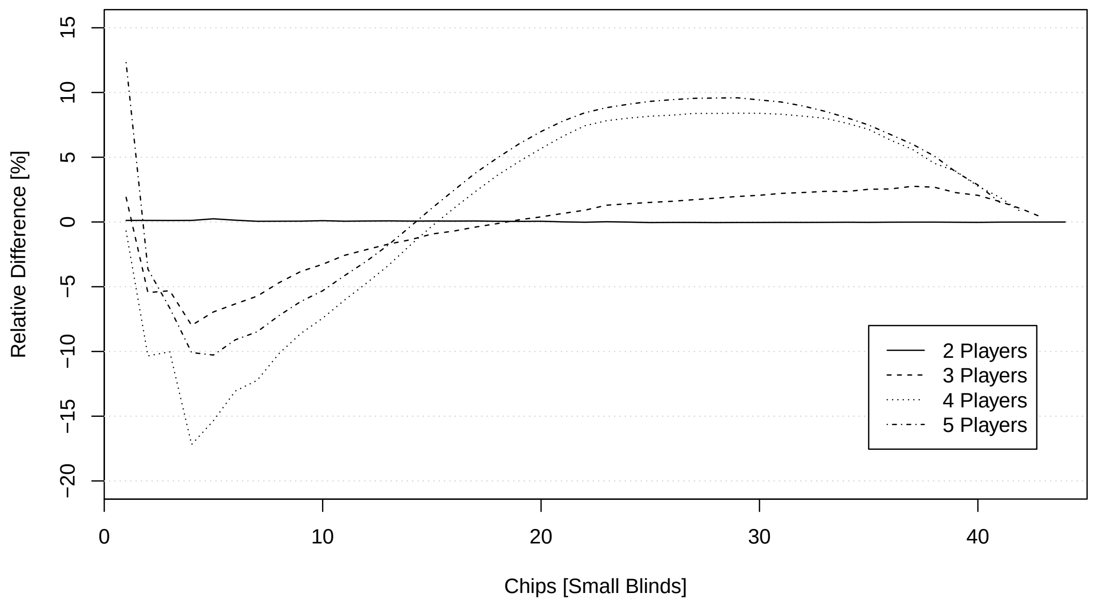

Figure 7.6: Relative difference between PI-CFR$^+$ winning probabilities and the linear approximation.

chips. This situation benefits players with large chip stacks, who can employ a very aggressive strategy to force their risk averse opponents out of pots.[2]

Figure 7.6 shows the relative signed deviation of the PI-CFR$^+$ winning percentages from the linear approximation. The biggest relative deviation can be seen at a stack size of 4 SBs and 4 remaining players. The winning percentage in the PI-CFR$^+$ solution in this case is 7.36%, which is 17.2% lower than predicted by the linear approximation (4/45 = 8.89%).

7.5 Absolute Payoff Deviation

Figures 7.2, 7.3, 7.4 from section 7.2 show the difference in mean payoffs and provide a good overview of model bias depending on the stack size, but it is important to remember that positive and negative errors in the estimates cancel each other out in these figures. They are therefore not a good measure for the overall accuracy of the estimates.

For reference we also provide figures 7.7, 7.8, 7.9, showing the mean absolute deviation between the model estimates and the payoff values in the PI-CFR$^+$ solution over all tournament states.

[2]This scenario is of considerable importance for human players when learning correct tournament play. The concept is widely known as “abusing the bubble” in the poker community.

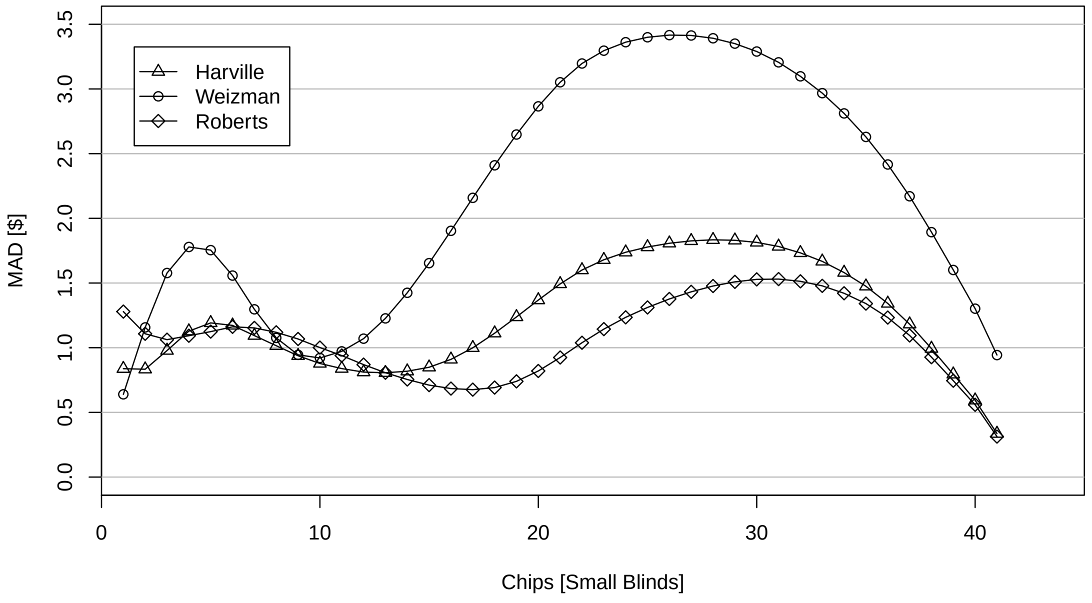

Figure 7.7: Mean Absolute Deviation for 5-player states.

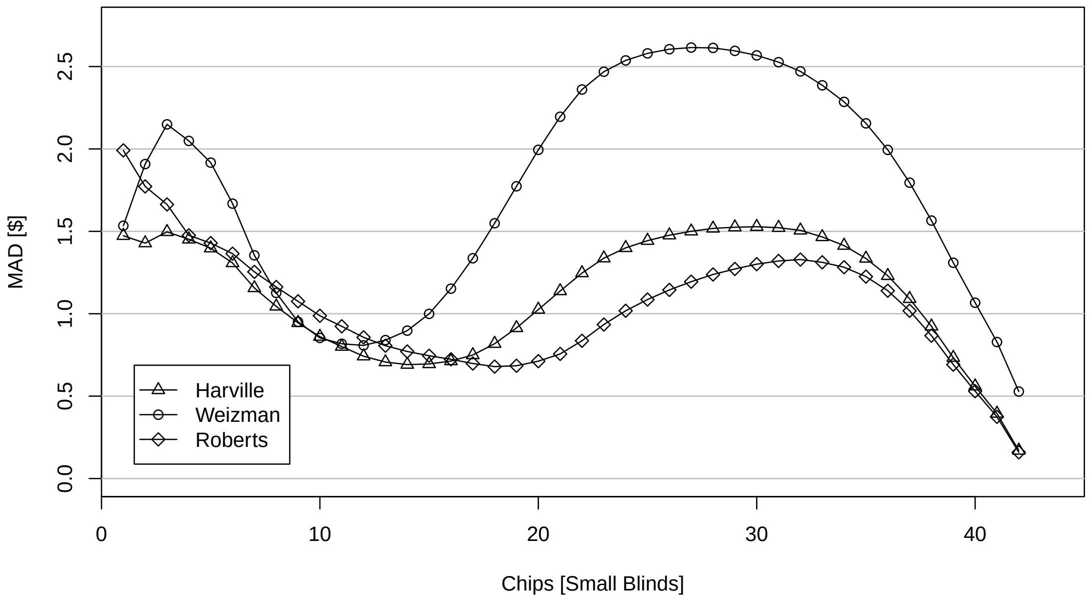

Figure 7.8: Mean Absolute Deviation for 4-player states.

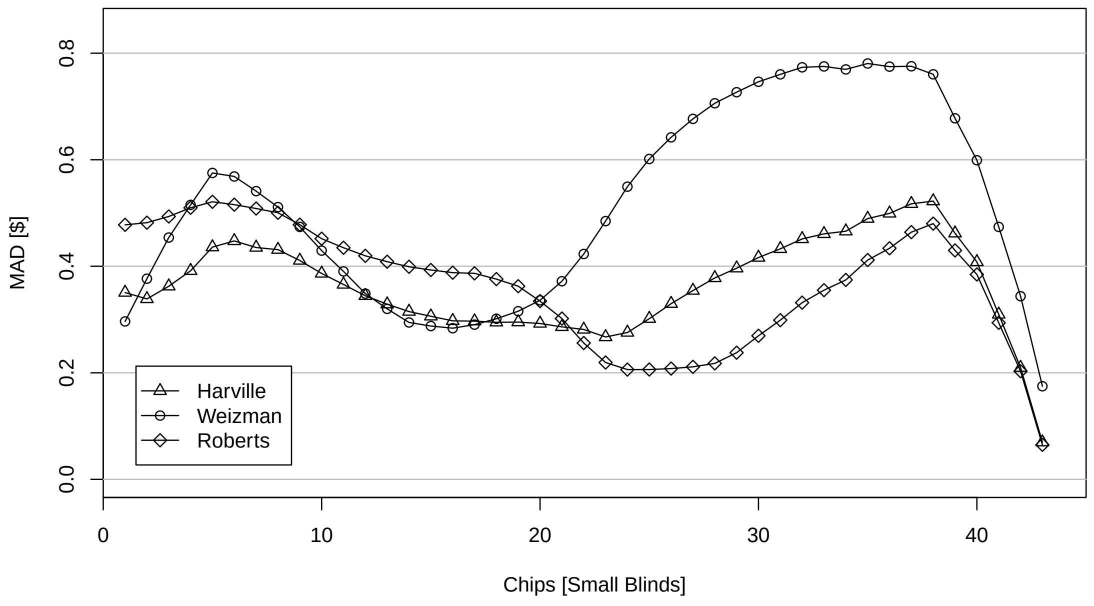

Figure 7.9: Mean Absolute Deviation for 3-player states.

CHAPTER 8

Strategy Evaluation

Aside from the statistical analysis of the equity estimates in the last chapter, it is also interesting to analyze the difference in game performance between strategy profiles depending on the equity model used to create them.

In this chapter, we will first compare the performance of the strategies against a theoretical worst case "*nemesis*" opponent. In the second part, we will run tournament simulations for players using different strategy profiles.

8.1 Equilibrium Quality

We used the *ex post* check algorithm, described in [GS09], to calculate ϵ and evaluate how well each of the strategy sets approximates a Nash equilibrium. For the PI-CFR$^+$ strategy profile, we found that starting in a random position of the starting state $x_{Start} = (9, 9, 9, 9, 9)$, a worst case opponent can increase his expected payoff by no more than an average of 0.0003\$ throughout the remainder of the tournament.

This is equivalent to an increase of $ROI = 0.0015\%$[1] from the 20.0\$ baseline. Over all states we get a maximum of $\epsilon = 0.00043\$$ with a mean value of 0.00021\$. So the PI-CFR$^+$ strategy set is indeed a very close Nash equilibrium approximation for our abstract tournament game.

Table 8.1 shows the same statistics for the other calculated models. The win-rate of a *nemesis* player in the starting state is considerably lower against the Roberts strategies ($ROI = 2.19\%$) than against the Malmuth-Harville ones ($ROI = 4.32\%$).

The results are similar for the average improvement of a *nemesis* player placed in a random tournament state, at $\epsilon = 0.262\$$ for Roberts and $\epsilon = 0.375\$$ for Malmuth-Harville.

[1]Return On Investment (ROI) is a popular performance evaluator for poker tournaments. For the calculation we assume each player contributes 1/5th of the 100\$ prize pool, so we use $ROI = \frac{payoff - 20\$}{20\$}$.

	Start State			**Global**	
	ROI[%]	**Mean[$]**	**Max[$]**	**Mean[$]**	**Max[$]**
Harville	4.32	0.864	1.069	0.375	3.802
Weitzman	15.20	3.040	3.527	1.288	4.897
Roberts	2.19	0.438	0.547	0.262	5.431
FGS-1	1.91	0.382	0.458	0.194	2.414
FGS-2	1.34	0.269	0.291	0.148	1.210
FGS-3	0.99	0.197	0.268	0.088	1.614
FGS-4	0.62	0.124	0.148	0.054	1.005
FGS-5	0.36	0.071	0.091	0.037	0.921
FGS-6	0.25	0.050	0.058	0.027	0.665
FGS-7	0.19	0.037	0.043	0.017	0.432
FGS-8	0.12	0.024	0.028	0.013	0.425
FGS-9	0.09	0.018	0.026	0.009	0.299
FGS-10	0.05	0.009	0.010	0.005	0.161

Table 8.1: *ex post* results summary

Malmuth-Weitzman is a distant third in both metrics, at $ROI = 15.2\%$ for the starting state and a global mean of $\epsilon = 1.288\$$.

Surprisingly, Roberts has a small percentage of big outliers and has the largest maximum $\epsilon = 5.431\$$ among the models, even higher than Malmuth-Weitzman at $\epsilon = 4.897\$$. This only concerns a very small fraction of states, however; Roberts has lower ϵ than Malmuth-Harville up to and including the 99th percentile.

8.2 Tournament Simulation

To determine the performance of two strategy profiles against each other, we decided to run Monte Carlo simulations of tournaments according to the rules of the abstract game, with players acting according to the different strategy profiles.

This approach is considerably easier to implement and much more flexible regarding the game settings that can be evaluated, compared to an adjustment of the tournament calculation.

Similar simulations were performed by a collaboration of tournament software developers[2] including this author. Results of two independent simulations indicated that strategies based on the Roberts model lose at approximately 1) $ROI = -0.36\%$ and 2) $ROI = -0.18\%$ against the Malmuth-Harville Model. The strategies for these simulations were calculated on-the-fly with a restricted strategy space as discussed in 5.4. As this is

[2]Discussions and results regarding these preliminary simulations can be found on the Poker Theory forum of *Two Plus Two Publishing*: `http://forumserver.twoplustwo.com/15/poker-theory/`

a relatively slow process, the sample sizes ranged between 10^5 - 10^6 tournaments per simulation.

Taking advantage of the limited number of tournament states in our abstract game, we can pre-calculate the strategies for the entire tournament. This means there are no expensive strategy calculations necessary during the Monte Carlo simulation, so the sample size can be increased by several magnitudes.

With on-the-fly calculations of the strategies, it took several seconds to simulate a single tournament for the old simulations, even using the restricted strategy space discussed in section 5.4. After pre-calculating the approximately 150,000 tournament states, our implementation is able to simulate around 500,000 tournaments per second.

We used an implementation of the Well44497b [PLM06] pseudorandom number generator to generate the random numbers required to deal cards, randomize decisions and apply the tiebreaker rule during the simulation.

8.3 Simulation Results

The tables in the following sections show the simulation results for one player using the single hand equilibrium strategies based on a specified equity model, simulated against the other players using the PI-CFR$^+$ strategies. Table 8.2 shows a summary of the simulation results.

Model	**ROI**[%]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
Harville	-0.998	18.49	20.43	22.13	20.59	18.36
Weitzman	-2.665	16.93	20.52	24.22	21.18	17.15
Roberts	-0.988	18.81	20.36	21.44	20.03	19.35

Table 8.2: Simulation summary for standard equity models

The tables are based on a 1×10^9 tournament sample for each of the starting positions, so a total of 5×10^9 simulated tournaments per equity model. The standard deviation for the mean payoffs over a positional sample of 1×10^9 tournaments is in the order of ± 0.0006\$.

We can see from tables 8.3 and 8.5 that the performance of the single hand strategies based on Malmuth-Harville and Roberts against the PI-CFR$^+$ solution, is virtually identical at approximately -1.0% of the baseline payoffs. The Malmuth-Weitzman strategies in 8.4 perform considerably worse at -2.7%.

Among all three models, there is a common trend of finishing considerably less often than the "fair" 20% in the 1st and 5th places. This indicates that strategies based on these models are too risk averse. This is consistent with our earlier observation in 7.2 that all three models considerably under-estimate the expected value of large chip stacks.

Start	**Payoff[$]**	**1st[%]**	**2nd[%]**	**3rd[%]**	**4th[%]**	**5th[%]**
HJ	19.585	18.21	20.21	22.08	20.47	19.03
CO	20.215	18.96	20.85	22.39	20.41	17.39
BU	21.232	19.84	22.09	23.43	19.63	15.01
SB	20.286	18.89	20.93	22.81	20.29	17.08
BB	17.683	16.56	18.06	19.92	22.15	23.31
Avg.	19.800	18.49	20.43	22.13	20.59	18.36

Table 8.3: Simulation details for Malmuth-Harville

Start	**Payoff[$]**	**1st[%]**	**2nd[%]**	**3rd[%]**	**4th[%]**	**5th[%]**
HJ	19.274	16.66	20.36	24.18	21.00	17.80
CO	19.899	17.50	20.96	24.31	20.46	16.77
BU	20.932	18.49	22.20	25.15	19.38	14.78
SB	19.970	17.35	21.02	24.96	20.45	16.22
BB	17.262	14.67	18.08	22.51	24.60	20.14
Avg.	19.467	16.93	20.52	24.22	21.18	17.15

Table 8.4: Simulation details for Malmuth-Weitzman

Start	**Payoff[$]**	**1st[%]**	**2nd[%]**	**3rd[%]**	**4th[%]**	**5th[%]**
HJ	19.582	18.57	20.11	21.31	19.83	20.18
CO	20.211	19.23	20.79	21.80	20.16	18.02
BU	21.223	20.02	22.02	23.03	19.55	15.37
SB	20.285	19.14	20.88	22.24	20.07	17.66
BB	17.711	17.09	18.01	18.82	20.56	25.53
Avg.	19.802	18.81	20.36	21.44	20.03	19.35

Table 8.5: Simulation details for Roberts

8.3.1 Direct Matches

In the simulations discussed so far, we had a single player using the tested strategy playing against the remaining opponents using the PI-CFR$^+$ strategy.

Since Malmuth-Harville and Roberts performed similarly in our evaluation so far, we decided to also run a direct simulation of these two strategy sets against each other.

We simulated 1×10^8 tournaments for each seating configuration in the starting state $x_{Start} = (9, 9, 9, 9, 9)$. With two models we have 30 seating configurations[3] and get a total sample size of 3×10^9 simulated tournaments.

The results are shown in table 8.6. We can see that the Roberts model indeed values big stacks slightly higher and is less risk averse, resulting in the higher finishing percentages for first and fifth places. Interestingly the models are essentially breaking even against each other, Malmuth-Harville having a barely positive *ROI* of +0.009%.

Our result is in considerable contrast to the results of the preliminary simulations mentioned earlier, where Malmuth-Harville had an observed *ROI* in the order of +1% against Roberts. This preliminary simulation used a much smaller sample size, restricted strategy spaces and a slightly different tournament model with an increasing blind structure. We suspect the different payout structure of $R = (65\$, 35\$)$ to be the most important factor for the differing results.

We used the same procedure to simulate Malmuth-Harville against Malmuth-Weitzman and Roberts against Malmuth-Weitzman; these results are also shown in table 8.6. The Malmuth-Weitzman strategy set loses at a considerable rate in both cases, with Malmuth-Harville having a slightly better $ROI = 1.057\%$ compared to an $ROI = 0.956\%$ for Roberts.

Matchup	**ROI**[%]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
Harville	0.009	19.88	20.00	20.29	20.34	19.48
Roberts	−0.009	20.12	20.00	19.71	19.66	20.52
Harville	1.057	20.90	19.95	18.88	19.57	20.70
Weitzman	−1.057	19.10	20.05	21.12	20.43	19.30
Roberts	0.956	20.94	19.97	18.64	19.27	21.17
Weitzman	−0.956	19.06	20.03	21.36	20.73	18.83

Table 8.6: Simulation results for direct matches

[3]A total of $2^5 = 32$ ways to assign one of the two strategy sets to each of the players, minus the two configurations which have all players using the same strategy set.

8.3.2 FGS Simulation

Table 8.7 shows a summary of the simulation results for FGS based strategies with varying depth levels against the PI-CFR$^+$ strategy set. The detailed results are included in appendix B. Similar to the *ex post* results earlier, we observe that FGS provides a considerable improvement over the results of Malmuth-Harville against PI-CFR$^+$ (see table 8.2), even with a low depth parameter.

Calculations at depth 2−3 are feasible for “on-the-fly” analysis even on moderate hardware, and provide strategies that reduce Malmuth-Harville’s negative ROI of −1% by about half, to −0.526% and −0.475% respectively, in our game setting. It is noteworthy that the *ROI* of the PI-CFR$^+$ solution against FGS strategy decreases almost monotonially with increasing depth parameter, with FGS-6 to FGS-7 being the only exception.

This relatively steady convergence is not shared by the finishing probabilities and the underlying strategies. This can be observed, for instance, in the finishing percentage for 1st or 5th place in table 8.7, these percentages move back and forth considerably as the depth increases[4].

Model	**ROI**[%]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
FGS-1	−0.695	19.03	20.31	21.25	20.68	18.72
FGS-2	−0.526	19.56	20.05	20.50	20.18	19.71
FGS-3	−0.475	19.79	20.00	20.05	20.02	20.14
FGS-4	−0.269	19.66	20.11	20.41	20.08	19.74
FGS-5	−0.178	19.72	20.05	20.46	20.25	19.53
FGS-6	−0.119	19.93	20.03	20.01	20.08	19.95
FGS-7	−0.126	20.02	20.01	19.81	19.93	20.23
FGS-8	−0.060	19.89	20.03	20.18	19.99	19.91
FGS-9	−0.039	19.94	20.03	20.08	19.99	19.96
FGS-10	−0.023	20.02	20.00	19.93	19.98	20.07

Table 8.7: Simulation summary for FGS

[4]We reiterate that the sampling error for the simulations is essentially negligible because of the big sample size. With a sample of 5×10^9 simulated tournaments for each entry of table 8.7 the standard deviation for the mean finishing percentages is in the order of ±0.00057 percentage points in this case.

CHAPTER 9

Conclusion

9.1 Summary

In this thesis we have discussed popular heuristics for estimating payoffs in poker tournaments and state-of-the-art methods such as CFR$^+$ and PI-FP for solving large stochastic games of imperfect information. We have also provided improved algorithms for calculating the Malmuth-Harville and Malmuth-Weitzman heuristics, reducing the runtime complexity from $O(n!)$ to $O(n \times 2^n)$ by utilization of a 2^n sized cache.

Using these methods, we calculated a Nash equilibrium approximation for a jam/fold tournament abstraction with 149,985 tournament states and 5.8×10^8 information sets, finding a set of strategies that is exploitable by no more than $\epsilon = 0.00043\%$ of the prize pool in any tournament state. This is to our knowledge the biggest tournament abstraction ever calculated in this way.

Using the payoffs of this tournament solution as a reference, we then evaluated the accuracy of popular heuristics. One of our most important findings is that the Malmuth-Harville and Roberts heuristics both considerably underestimate payoffs for large chip stacks. In the case of Malmuth-Harville this was already relatively well known, but the Roberts model was explicitly designed in an attempt to address this shortcoming. It is an interesting result that Roberts also shares this tendency, although to a reduced extent.

In the statistical evaluation of model estimates against the PI-CFR$^+$ payoffs, the *de facto* standard Malmuth-Harville outperformed Roberts consistently across all evaluated metrics. The Malmuth-Weitzman model performed considerably worse than the other two models and should, in our opinion, be disregarded for practical applications.

We also found that an assumption shared by all heuristic models, winning probabilities being proportional to chip stacks, does not hold true in our game setting. This is the case even after we account for positional factors. We speculate that this higher-than-proportional winning percentage for large chip stacks may account for a considerable part of the bias in the payoff estimates.

Positional factors appear to contribute around $2 - 3$ percentage points to the MAPD of the estimates. This value can serve as a rough estimate of the lower bound achievable by an equity model working without positional information.

Results for our *ex post* evaluation showed that a worst case opponent can win at $ROI = 4.32\%$ against Malmuth-Harville based strategies and only 2.19% against Roberts. These values are on the lower range of our expectations, and Roberts substantially outperforming Malmuth-Harville is especially surprising considering our earlier results in the statistical evaluation.

In terms of simulation results, our findings are in line with preliminary simulations, with the exception of the strategies based on the Roberts model. These do perform better than indicated by previous simulations and are essentially break-even in a direct matchup against Malmuth-Harville, with both models losing at an almost identical rate of approximately $ROI = -1.0\%$ against the PI-CFR$^+$ strategy set. Considering our simulation results and the the *ex post* evaluation, the Roberts heuristic does look like a viable alternative for practical use.

Evaluation of strategies based on the FGS algorithm found a substantial improvement over the standard heuristics, even when using relatively low values for the depth parameter. The accuracy appears to increase consistently with the depth parameter.

9.2 Restrictions and Further Work

It is important to keep in mind that our evaluation results are based on a specific tournament game. Most importantly, although the distribution of payouts as $R = (50\%, 30\%, 20\%)$ is very popular in STT games, there are several other payout structures in widespread use. It is not clear if our results are specific to this particular payoff structure or if our findings apply to a broader setting. Additional evaluations using other payoff structures would be interesting to provide clarification.

Regarding the FGS findings, it should be noted that we used a variant of VI-CFR$^+$ to calculate the FGS strategy sets. This is equivalent to FGS with an unrestricted strategy space, while most publicly available implementations of FGS use restricted strategy spaces to speed up the calculation. For this reason, the results may not be entirely representative of these implementations. We do not expect a large difference in results, but FGS with a restricted strategy space should be considered for future evaluations.

We provided a detailed analysis about the shortcomings of the current heuristics. Hopefully, this information, along with the provided PI-CFR$^+$ finishing probabilities for our game, will prove useful in the development of more accurate models.

APPENDIX A

Java Listings

```java
/**
 * @param x chip stack for each player
 * @return finishing distribution matrix P
 */
public static double[][] getHarvilleMatrixFast(double[] x){

    double[][] p = new double[x.length][x.length];
    double[] w = new double[(0x1 << (x.length))];
    w[0] = 1;

    double r, t;
    for (int f = 1; f <= x.length; f++)
        for (int ns = 1; ns < w.length; ns++)
            if (Integer.bitCount(ns) == f) {
                r = 0;
                for (int i = 0; i < x.length; i++)
                    if (((0x1 << i) & ns) == 0)
                        r += x[i];
                for (int i = 0; i < x.length; i++)
                    if (((0x1 << i) & ns) != 0) {
                        t = x[i] / (x[i] + r) * w[ns ^ (0x1 << i)];
                        p[i][f - 1] += t;
                        w[ns] += t;
                    }
            }

    return p;
}
```

Listing A.1: Minimalistic Java implementation of our $O(n \times 2^n)$ Malmuth-Harville variant, corresponding to algorithm 2.1

```java
    /**
     * @param x chip stack for each player
     * @return finishing distribution matrix P
     */
    public static double[][] getWeitzmanMatrixFast(double[] x){

        double[][] p = new double[x.length][x.length];
        double[] w = new double[(0x1 << (x.length))];
        w[w.length - 1] = 1;

        double d, t, b;
        for (int r = x.length; r > 0; r--)
            for (int ns = 0; ns < w.length; ns++)
                if (Integer.bitCount(ns) == r) {
                    d = 0;
                    for (int i = 0; i < x.length; i++)
                        if (((0x1 << i) & ns) == 0)
                            d += x[i];
                    d /= r;

                    b = 0;
                    for (int i = 0; i < x.length; i++)
                        if (((0x1 << i) & ns) != 0)
                            b += 1.0 / (x[i] + d);

                    for (int i = 0; i < x.length; i++)
                        if (((0x1 << i) & ns) != 0) {
                            t = 1.0 / (b * (x[i] + d)) * w[ns];
                            p[r - 1][i] += t;
                            w[ns ^ (0x1 << i)] += t;
                        }
                }

        return p;
    }
```

Listing A.2: Minimalistic Java implementation of our $O(n \times 2^n)$ Malmuth-Weitzman variant, corresponding to algorithm 2.2

APPENDIX B

Detailed FGS Results

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.646	18.88	20.04	20.96	19.98	20.14
CO	20.278	19.64	20.67	21.28	20.18	18.23
BU	21.290	20.38	21.99	22.51	20.18	14.93
SB	20.341	19.17	20.91	22.42	21.20	16.31
BB	17.751	17.09	17.96	19.08	21.86	24.01
Avg.	19.861	19.03	20.31	21.25	20.68	18.72

Table B.01: Simulation details for FGS-1

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.673	19.31	19.79	20.40	19.81	20.69
CO	20.312	20.26	20.35	20.39	19.20	19.81
BU	21.324	21.01	21.67	21.57	19.53	16.21
SB	20.379	19.70	20.68	21.62	20.77	17.22
BB	17.786	17.52	17.75	18.50	21.59	24.64
Avg.	19.895	19.56	20.05	20.50	20.18	19.71

Table B.02: Simulation details for FGS-2

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.682	19.42	19.79	20.18	20.27	20.34
CO	20.319	20.41	20.34	20.07	19.24	19.95
BU	21.334	21.38	21.53	20.94	18.89	17.27
SB	20.391	20.07	20.57	20.93	20.09	18.34
BB	17.799	17.69	17.76	18.14	21.60	24.82
Avg.	19.905	19.79	20.00	20.05	20.02	20.14

Table B.03: Simulation details for FGS-3

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.725	19.37	19.89	20.37	20.05	20.32
CO	20.361	20.20	20.50	20.57	19.93	18.81
BU	21.376	21.11	21.69	21.57	19.39	16.24
SB	20.430	20.03	20.62	21.16	19.90	18.30
BB	17.838	17.61	17.87	18.38	21.12	25.03
Avg.	19.946	19.66	20.11	20.41	20.08	19.74

Table B.04: Simulation details for FGS-4

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.742	19.47	19.79	20.36	19.88	20.50
CO	20.378	20.31	20.39	20.52	19.92	18.86
BU	21.389	21.08	21.68	21.74	20.07	15.44
SB	20.447	19.89	20.64	21.55	20.74	17.18
BB	17.866	17.85	17.73	18.12	20.64	25.67
Avg.	19.964	19.72	20.05	20.46	20.25	19.53

Table B.05: Simulation details for FGS-5

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.751	19.66	19.77	19.95	19.75	20.86
CO	20.389	20.47	20.40	20.17	19.71	19.25
BU	21.403	21.38	21.64	21.12	19.68	16.19
SB	20.462	20.12	20.63	21.06	20.66	17.53
BB	17.876	18.03	17.71	17.73	20.60	25.93
Avg.	19.976	19.93	20.03	20.01	20.08	19.95

Table B.06: Simulation details for FGS-6

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.750	19.65	19.80	19.92	20.11	20.52
CO	20.387	20.55	20.39	19.97	19.50	19.59
BU	21.401	21.50	21.59	20.87	19.29	16.75
SB	20.459	20.31	20.56	20.67	20.10	18.36
BB	17.877	18.09	17.71	17.60	20.65	25.95
Avg.	19.975	20.02	20.01	19.81	19.93	20.23

Table B.07: Simulation details for FGS-7

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.764	19.61	19.79	20.12	19.90	20.58
CO	20.399	20.42	20.41	20.34	19.85	18.99
BU	21.414	21.30	21.63	21.37	19.57	16.13
SB	20.472	20.15	20.58	21.12	20.09	18.06
BB	17.891	17.95	17.75	17.94	20.54	25.81
Avg.	19.988	19.89	20.03	20.18	19.99	19.91

Table B.08: Simulation details for FGS-8

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.767	19.67	19.77	20.00	19.71	20.84
CO	20.406	20.51	20.39	20.17	19.70	19.23
BU	21.419	21.35	21.64	21.27	19.76	15.99
SB	20.475	20.14	20.59	21.12	20.37	17.77
BB	17.894	18.01	17.73	17.84	20.42	26.00
Avg.	19.992	19.94	20.03	20.08	19.99	19.96

Table B.09: Simulation details for FGS-9

Start	**Payoff**[$]	**1st**[%]	**2nd**[%]	**3rd**[%]	**4th**[%]	**5th**[%]
HJ	19.770	19.73	19.76	19.88	19.74	20.89
CO	20.409	20.58	20.37	20.05	19.68	19.33
BU	21.421	21.49	21.59	20.99	19.54	16.38
SB	20.480	20.19	20.59	21.02	20.50	17.69
BB	17.897	18.09	17.70	17.70	20.43	26.08
Avg.	19.995	20.02	20.00	19.93	19.98	20.07

Table B.010: Simulation details for FGS-10

List of Figures

List of Tables

List of Algorithms

Acronyms

Bibliography

[ACP] ACPC. The Annual Computer Poker Competition webpage. http://www.computerpokercompetition.org/. Accessed: 2015-11-20.

[BBJT15] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[Bro51] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[CA06] B. Chen and J. Ankenman. *The Mathematics of Poker.* ConJelCo LLC, 2006.

[FL98] Drew Fudenberg and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

[Gan15] Sam Ganzfried. My reflections on the first man vs. machine no-limit texas hold'em competition. *arXiv preprint arXiv:1510.08578*, 2015.

[GS08] Sam Ganzfried and Tuomas Sandholm. 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 Multiagent Systems - Volume 2*, AAMAS '08, pages 919–925, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[GS09] Sam Ganzfried and Tuomas Sandholm. Computing equilibria in multiplayer stochastic games of imperfect information. In *IJCAI*, pages 140–146, 2009.

[Har73] David A Harville. Assigning probabilities to the outcomes of multi-entry competitions. *Journal of the American Statistical Association*, 68(342):312–316, 1973.

[JBL$^+$12] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 837–846. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[Joh07] Michael Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007.

[JWBZ11] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *IJCAI*, volume 11, pages 258–265, 2011.

[LZ06] Michael Littman and Martin Zinkevich. The 2006 aaai computer poker competition. *ICGA Journal*, 29(3):166, 2006.

[Mic66] Donald Michie. Game-playing and game-learning automata. *Advances in programming and non-numerical computation*, pages 183–200, 1966.

[MS07] Peter Bro Miltersen and Troels Bjerre Sørensen. A near-optimal strategy for a heads-up no-limit texas hold'em poker tournament. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 191. ACM, 2007.

[PLM06] François Panneton, Pierre L'ecuyer, and Makoto Matsumoto. Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software (TOMS)*, 32(1):1–16, 2006.

[Pou92] William Poundstone. Prisoner's dilemma: John von neuman, game theory, and the puzzle of the bomb, 1992.

[Put05] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2005.

[Rob11] Ben Roberts. A new algorithm for the approximation of icm equities in tournament poker, 2011.

[RW11] Jonathan Rubin and Ian Watson. Computer poker: A review. *Artificial Intelligence*, 175(5):958–987, 2011.

[Sch01] Jonathan Schaeffer. A gamut of games. *AI Magazine*, 22(3):29, 2001.

[Skl99] David Sklansky. *The theory of poker.* Two Plus Two Publishing LLC, 1999.

[Tam14] Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.

[VNM44] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior.* Princeton university press, 1944.

[ZJBP08] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1729–1736. MIT Press, Cambridge, MA, 2008.