Robust Strategies and Counter-Strategies: From Superhuman to Optimal Play

by

Michael Bradley Johanson

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

Games have been used as a testbed for artificial intelligence research since the earliest conceptions of computing itself. The twin goals of defeating human professional players at games, and of solving games outright by creating an optimal computer agent, have helped to drive practical research in this field. Deep Blue defeating Kasparov at chess and Chinook solving the game of checkers serve as milestone events in the popular understanding of artificial intelligence. However, imperfect information games present new challenges and require new research. The Abstraction-Solving-Translation procedure for approaching such games involves abstracting a game down to a tractable size, solving the abstract game to produce a strong abstract strategy, and translating its decisions into the real game as needed. Related challenges include principled evaluation of the resulting computer agents, and using opponent models to improve in-game performance against imperfect adversaries. The papers presented in this thesis encompass the complete end-to-end task of creating strong agents for extremely large games by using the Abstraction-Solving-Translation procedure, and we present a body of research that has made contributions to each step of this task. We use the game of poker as a testbed domain to validate our research, and present two milestone accomplishments reached as a result: the first victory of a computer agent over human professionals in a meaningful poker match, and the first solution to any imperfect information game played competitively by humans.

Preface

This thesis is a work written by myself, Michael Bradley Johanson. In a paper-based thesis, each chapter consists of a previously published paper, which was written in collaboration with my coauthors. Each chapter will begin with a brief introduction that places the paper in context among the other papers, and identifies my contributions to the work.

Journal papers must also be described in this preface. Chapter 9 of this thesis has been published as M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up Limit Texas Hold'em is Solved", *Science*, vol. 347, number 6218, 145–149, January 2015. Tammelin invented the CFR+ algorithm and streaming compression technique used in the paper. Our implementation of CFR+, designed for high performance clusters, was written and tested by Burch and myself. I was responsible for tuning parameters to find an acceptable three-way tradeoff between solution quality, disk and RAM memory required, and computation time. I also ran the three-month long computation and collected the empirical results. Bowling led our decade-long effort to solve the game and was responsible for composing the manuscript. Burch and I assisted with editing and formatting.

Losing is Fun!

– Traditional Dwarf Fortress encouragement

Acknowledgements

The years I've spent as a grad student have been my happiest, due to my excellent group of friends and colleagues. I owe my heartfelt thanks to:

- My supervisor, **Michael Bowling**, for his guidance, support, and friendship over the last twelve years.
- My family, **Brad, Sue, and Jeff Johanson**, and my partner, **Maren Wilson**, for their love and support.
- My friends in the Computer Poker Research Group, from whom I've learned so much: **Martin Zinkevich, Duane Szafron, Rob Holte, Jonathan Schaeffer, Chris Archibald, Darse Billings, Viliam Lisý, Nolan Bard, Neil Burch, Trevor Davis, Richard Gibson, John Hawkin, Kevin Waugh, Nick Abou Risk, Josh Davidson, Morgan Kan, Parisa Mazrooei, Dustin Morrill, Dave Schnizlein, and Bryce Paradis**.
- My poker research colleagues: **Oskari Tammelin, Eric Jackson, Mihai Ciucu, Sam Ganzfried, Noam Brown, and Tuomas Sandholm**.
- My dear friends **Richard Valenzano and Jacqueline Smith**.
- **Anna Koop, Claire McGuigan, and Steph Herbert**, for our many Thursday afternoons drinking and thesis writing at The Druid.
- **Shayna Bowling, Leah Hackman, Joel Koop, Kit Chen, Barry Gergel, Sheehan Khan, John McGuigan, Julie-Ann Sarah, Diane Kan, Andrew Butcher, Curtis Onuczko, Simone Casavant, Jeff Ryan, Jeff Siegel, and William Thorne** for our time spent gaming and curling.
- Friends a few steps ahead of me in grad school, who I admired and wanted to follow: **Paul Berube, Colin Cherry, Dan Lizotte, John Arnold, Brian Tanner, Alona Fyshe, David Thue, Marc Bellemare, Marc Lanctot, Martha White, Joel Veness**, and most of all, **Jess Enright**.
- Finally, my friends from Happy Harbor: **Jay Bardyla, Melissa Cilliers, Andrea Clarke, Spencer Clarke, Rudi Gunther, Alena Helgeson, Craig Howrie, Greg Lekivetz, Wes Montey, Sanford O'Donnell, Shawna Roe, Dan Schneider, Phil Scott, and Troy Voong**.

And many more. Thank you for being part of my life, and for your role in this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The creation of artificially intelligent game-playing computer programs has been a motivating goal and a research testbed throughout the history of computing science and artificial intelligence. In the earliest days of the study of computation, Babbage wrote out the plans for his Analytical Engine, and then sought a test domain in which to demonstrate its power. He decided on "games of purely intellectual skill", such as tic-tac-toe, checkers, and chess [6, p. 465]. While his contemporaries believed that human reason was required to play these games of skill, Babbage realized that computational power alone would allow for game-tree search capable of playing such games optimally[1].

Alan Turing, a father of both computing science and artificial intelligence, was also intrigued by games and game-playing programs. Before the creation of the first computers, he had designed an algorithm for playing chess, which he would execute by hand in games against friends [42, p. 44]. Later, in his description of the Turing test for investigating whether or not an entity should be considered to be intelligent, he offered chess problems as an example of questions that might be asked [98].

John von Neumann, a pioneer in game theory and computing science, also studied this connection between games, intelligence, and computation. His earlier work in game theory sought to use games and mathematically sound strategies as a model for understanding real-life behaviours. And while game theory can be used to formalize "perfect information" games such as chess and checkers, as considered by both Babbage and Turing, von Neumann was more interested in understanding "imperfect information" games such as poker, which involve deceptive actions such as bluffing. In a conversation recounted by Bronowski, von Neumann noted this distinction [20]: "Real life is not like that. 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." After launching the field of game theory and connecting it to the mathematical framework of linear program-

[1] Babbage also realized the entertainment and financial possibilities of creating an "automaton" for such games. In the pages following his description of game tree search, he presented detailed plans for a travelling exhibition, and wondered if it might help him acquire the funds necessary to complete his Analytical Engine.

ming, von Neumann devoted his attention to the study of computation, becoming a founding father of the nascent field of computing science.

While these early pioneers saw the potential of computing for producing "intelligent" behaviour, the necessary computational hardware did not yet exist to realize their dreams. This deficiency has now been addressed: powerful desktop computers are ubiquitous and inexpensive, researchers have access to increasingly powerful supercomputers and custom hardware, and individual programmers can rent time on clusters such as Amazon Web Services (AWS). Combined with algorithmic advances for efficiently using computation to reason about decisions in ever-larger problem domains, it is now possible to create computer programs that can outperform humans in these "games of purely intellectual skill". Researchers and programmers now race to be the first to create programs capable of defeating human champions in games, and the resulting victories have served as milestone events in the popular understanding of artificial intelligence: the checkers-playing program Chinook becoming the first program to win a world championship against humans [86], IBM's Deep Blue achieving its famous victory over Kasparov in chess [43], and IBM's Watson defeating Jennings and Rutter on Jeopardy! [29]. Computers have also surpassed human champions at other games such as backgammon [96], othello [22], and Scrabble [88]. Today, this line of research continues in efforts to create superhuman programs for poker, go, Starcraft, billiards, Robocup (robot soccer), classic Atari 2600 games, and many others, including the General Game Playing challenge [31], in which programs must play arbitrary and novel games after being given the rules just before a match begins, without intervention by their human programmers.

The connection between the earlier work by Babbage, Turing and von Neumann to this modern research effort is central to the study of artificial intelligence. **We aspire to create general programs that can reason about their environment and the other agents sharing it, and make decisions so as to achieve their goals**, just as human intelligence allows each of us to plan, act, negotiate, and succeed in our own lives.

When we aspire to create artificially intelligent agents for complex real world tasks, games provide a tractable gradient that we can follow towards this goal: a series of increasingly complex domains in which state-of-the-art techniques are inapplicable or intractable, requiring us to develop and evaluate new approaches. The challenges provided by games can scale in many dimensions. Games can increase in size, such as in the number of decision points and actions that players must consider. They can have perfect information such that all players can observe the exact game state, or introduce imperfect information such that some actions that take place are not observable by one or more players. They can have only deterministic actions, or introduce stochastic elements that make the state transitions unpredictable. A player's utility at the end of the game might be a simple win-lose-or-tie outcome, or might be a continuous value that the player aims to maximize. And finally, the other agents in a game can affect its difficulty in many ways. Are there no other agents sharing the space, or one, or many? Are they adversarial or cooperative? How capable are they at planning and executing their own actions to achieve their

goals?

As computers have now surpassed human performance in a number of perfect information, deterministic, two-player, zero-sum games, such as chess and checkers, and a small number of games with other qualities such as backgammon and Scrabble, we are now in an exciting period when researchers are following this gradient along several dimensions towards more complex games, and developing new artificial intelligence techniques capable of handling their larger scales and novel qualities. As we continue to expand our efforts into new and diverse challenges, we are approaching the scope of real world problems in which **agents** – by which term we refer to robots, computer programs, or people – must cope with many forms of uncertainty: uncertainty about the other agents acting in the domain such as their number, motivations, or abilities; uncertainty about the world due to their imperfect senses and knowledge; and uncertainty about how the future will unfold due to stochastic events. Further, real-world tasks rarely have the win-lose-or-tie outcomes, and instead may have real-valued outcomes such as maximizing an expected income, or minimizing risk, or minimizing the time or financial cost to achieve a goal.

In this thesis, we will explore the practical and theoretical challenges involved in creating strong computer agents for the large and complex games that humans find challenging and enjoyable. This work is built upon the mathematical foundation of **game theory**.

1.1 Game Theory

Game theory, developed by John von Neumann, studies the interaction of agents in a shared environment. While the term "game" brings to mind recreational games played for entertainment and friendly competition, in this setting it also refers to models of serious interactions such as auctions, negotiations, homeland security [92], and cold war politics [69].

Game theory investigates the question: given knowledge about a domain (*e.g.* its rules and goals), how should a rational agent act so as to maximize its utility, and what behaviour can be expected of other rational agents? Game theory provides several answers to this question, called **solution concepts**, which depend on what information is known or assumed about the other agents in the space. For example, if the behaviour of the other agents is completely known *a priori*, then game theory suggests that a perfectly rational agent should use a **best response** strategy, which exploits this knowledge of the other agents so as to attain the highest possible expected utility. If nothing is known about the other agents, but we wish to perform well under the assumption that they are each perfectly rational, then game theory may suggest the use of a **Nash equilibrium** strategy, in which each agent simultaneously uses a best response to the other agents in the game. In the case of two-player zero-sum games, a Nash equilibrium strategy (in this setting equivalent to a **minimax** strategy, and also called an **optimal** strategy) is particularly compelling. In this setting, a Nash equilibrium is guaranteed to do no worse than

tie (on expectation) against any opponent, including a worst-case opponent who knows every detail of the strategy *a priori*, and has unlimited computational power with which to compute a best response. Computing a Nash equilibrium is called **solving** a game: this technical term is used in exactly the same sense that one might "solve" a mathematical equation, such as by "solving for x".

The Nash equilibrium concept is particularly compelling as the foundation of a strong computer agent. In the two-player zero-sum setting, such a strategy would be guaranteed not to lose against any human or computer adversary, and may win by a small amount if the opponent is flawed. In practice, our ability to construct computer agents that use Nash equilibrium strategies is limited by our ability to solve large games. Fortunately, there are well-known algorithms for computing or approximating Nash equilibrium strategies in two-player zero sum games, and they can be precomputed without requiring any *a priori* knowledge about the opponent.

Over time, as we have discovered new and more efficient algorithms for solving games, we have increased the size of game that can be tractably solved. Many simple games, such as tic-tac-toe or rock-paper-scissors, can be solved in one's head through simple reasoning. Some small games, such as Kuhn poker (a three-card, one-bet poker game) [60], can be solved algebraically on paper. As we begin to consider larger games, we very quickly need to rely on powerful computational hardware and efficient algorithms to perform this reasoning for us. For example, a slightly larger "toy" poker game called Leduc hold'em (a six-card, two-round, four-bet poker game) [90] is easily solvable using modern algorithms and trivial hardware, but would be intractable using the algorithms known to von Neumann even with modern hardware.

In Figure 1.1, we examine our ability to solve ever-larger games over time by considering the imperfect information and stochastic game of poker, which has been a common research testbed for artificial intelligence researchers for more than fifteen years [12; 13; 89; 35], particularly after the founding of the **Annual Computer Poker Competition (ACPC)** in 2006 [65], and of interest to game theorists since von Neumann's seminal work in 1947 [100]. This figure presents the sizes of the largest-known poker games solved over time. The shaded regions indicate three families of game-solving algorithms: sequence-form linear programming (SFLP), Counterfactual Regret Minimization (CFR), and a recent extension of CFR known as CFR+. Over the last ten years, through algorithmic advances and the use of increasingly powerful computational hardware, there has been a seven order of magnitude increase in the size of imperfect information game that can be tractably solved. The first labelled datapoint, Rhode Island hold'em, is a synthetic game[2] first solved by Gilpin and Sandholm using sequence-form linear programming [35]. Ten years later, my colleagues and I have now solved heads-up limit Texas hold'em (HULHE), the simplest form of poker played by humans for high

[2]We will use the term "toy" to refer to games of trivial size, and "synthetic" to refer to games invented by researchers as a problem testbed. Tic-tac-toe is a trivial game played by humans and is thus a toy game but not a synthetic one; Rhode Island hold'em, created by Shi and Littman [89] as a research testbed with 10^6 decision points, is a synthetic game but not a toy game. A "human-scale" game is a nontrivial game that is played competitively by humans.

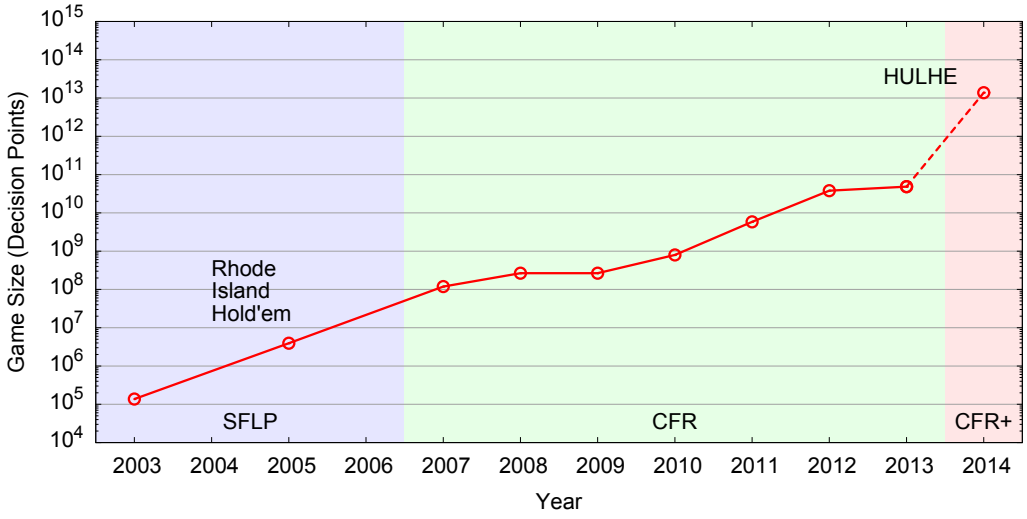


Figure 1.1: The increasing sizes of solved imperfect information games over time.

stakes[3], and the first imperfect information game played competitively by humans to be solved [17]. The unlabelled datapoints in Figure 1.1 are synthetic, simplified versions of HULHE, solved by researchers in an attempt to create strong HULHE strategies.

In addition to heads-up limit Texas hold'em, a small number of perfect information games played by humans have been solved, such as checkers [85] and awari [82]. However, solving such human-scale games has thus far required not only efficient algorithms and powerful computational hardware, but also considerable engineering effort. And while this combination has allowed us to solve perfect information games as large as checkers with 10^{20} decision points, or imperfect information games as large as heads-up limit Texas hold'em poker with 10^{13} strategically distinct decision points – an effort that required 900 CPU-years of computation – many other games played by humans are far, far larger. For example, the natural next variant of poker to consider after heads-up limit Texas hold'em is heads-up *no-limit* Texas hold'em, in which the players may make bets of any size instead of a fixed size. However, the no-limit variant, as played by researchers in the Annual Computer Poker Competition, has 10^{160} strategically distinct decision points [48, Table 6]. Solving this complete game directly appears entirely infeasible at this time[4].

[3]In fact, HULHE was the variant of poker played in the highest-stakes poker cash games ever played, in a series of matches between the banker Andy Beal and a team of poker pros. Their series of matches was described by Michael Craig in his book, "The Professor, the Banker, and the Suicide King: Inside the Richest Poker Game of All Time" [27].

[4]Indeed, even simply recording a strategy for this game by representing the probability of each action with one byte would require 5×10^{135} yottabytes of storage [48, p. 12]. Of course, this immense size represents a brute-force approach by representing each action individually, and does not preclude the future development of a lossy approximation that is sufficiently close to optimal, or an alternate (perhaps functional) representation of a strategy that may require far less storage.

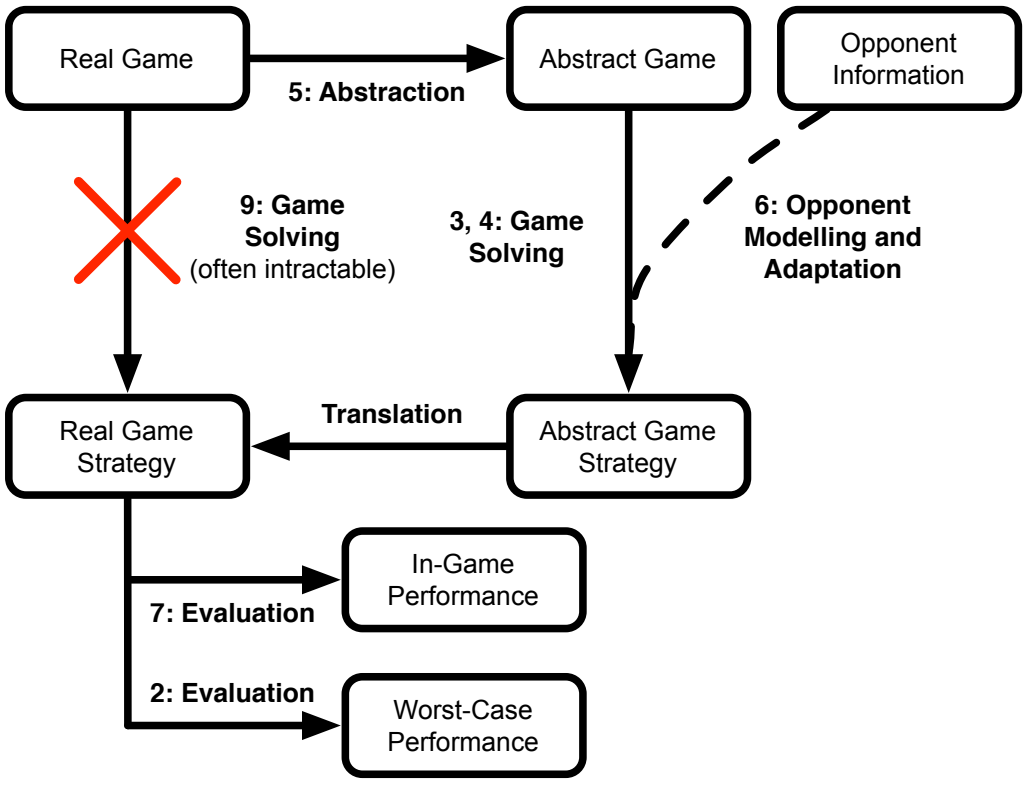


Figure 1.2: The Abstraction-Solving-Translation procedure. The numbers before each label indicate the chapters in which that arc will be discussed.

1.2 The Abstraction-Solving-Translation Procedure

Fortunately, it is not necessary to exactly solve a game in order to make good decisions or exceed human capabilities, or even to closely approximate optimal behaviour. In perfect information games such as chess, algorithms such as $\alpha - \beta$ search can consider only the current subgame that the player is in, and search many moves ahead by using a heuristic function to evaluate non-terminal nodes. While the use of a heuristic to explore a bounded horizon may mean that the program plays suboptimally, a fast, accurate heuristic and deep search may still be sufficient for superhuman performance.

In imperfect information games such as poker, it is much more difficult to reason about small subgames in this way and still obtain strong strategies. Instead, the dominant approach is to use the **Abstraction-Solving-Translation procedure** shown in Figure 1.2, which we will now explain, starting from the top left corner. If the game is too large to be tractably solved using the best available algorithms and computational hardware (an attempt to follow the downwards arc), then an alternative is to simplify the problem by creating a smaller, but strategically similar, **abstract game**. One way this is done is by finding decision points in the real game[5]

[5]For clarity, the term "real game" specifies the game defined by the rules, from which an abstract

that are similar, and merging them together in the abstract game, so that a player in the abstract game cannot distinguish them. We then use game theoretic algorithms to solve this abstract game, creating an **abstract game strategy**. Once computed, an agent can use this abstract strategy to play the real game, by **translating** portions of the abstract strategy back into the real game as needed. This approach has a long history in the computer poker community and is used by most of the top competitors in that domain [89; 11; 36; 46]. As Aumann describes, it is also the dominant approach for applying game theoretic reasoning to economics problems [5]: "The method of von Neumann and Morgenstern has become the archetype of later applications of game theory. One takes an economic problem, formulates it as a game, finds the game-theoretic solution, then translates the solution back into economic terms."

Under this paradigm we can increase the strength of our decision-making in two ways, involving both theoretical research and practical engineering. First, by developing new game solving algorithms that are more efficient in both space and time, we become able to solve larger, finer-grained abstract games that require less merging of decision points and thus are better models of the real game. Second, by developing new abstraction and translation techniques that preserve more of the strategically important structure during the abstraction process, we can also create abstract games that are better models of the real game. In practice, solving these "bigger" and "better" abstract games results in abstract strategies that are less suboptimal in the real game.[6]

To measure the progress that is being made, we must have some method of evaluating the agents that are created. There are two evaluation techniques that are commonly used: (1) observing an agent compete in games against humans or other computer agents, and (2) objectively measuring its suboptimality in the real game. Each of these evaluation methods, in-game performance and worst-case performance, offer distinct and useful feedback but present challenges that must be addressed through research and careful engineering. For example, all games have variance in their outcomes, and a large number of observed games may be necessary to obtain a statistically significant measurement of one player's edge over another. In high-variance games such as poker, humans may be unwilling to play the millions of games required to discern a small skill difference between two players. However, by exploiting properties of the game or our knowledge of a computer agent's strategy, we can compute lower variance, unbiased estimates of this difference. Likewise, measuring an agent's worst-case performance tells us how suboptimal it is compared to a Nash equilibrium strategy. This is a far easier computational task than computing a Nash equilibrium, and so may be feasible even when solving the game is infeasible. In HULHE, a direct, naive attempt to do so using the standard expectimax algorithm may still take decades of CPU time. By exploit-

game is derived.

[6]While this connection between better abstractions and stronger strategies is intuitive and useful in practice, it is unfortunately not supported by any theory. In fact, counter-examples called "abstraction pathologies" [103] have been discovered, where even a strict improvement to an abstraction can result in greater suboptimality in the real game. We will discuss these effects in Chapters 2 through 5.

ing knowledge of the game's structure, we can make this worst-case performance computation tractable and even convenient, and thus obtain a precise and objective measure of our agent's performance.

One final aspect of creating a strong computer agent is to consider what information is known about the other agents. While a Nash equilibrium strategy is a robust "defensive" strategy that is useful against any opponent, it will not take advantage of knowledge about the other agents in order to improve its performance. For example, we may have some *a priori* information about an adversary, such as observations of their earlier behaviour that demonstrates flaws or tendencies. We can use this information during the solving process to create a **counter-strategy** for our agent that will perform much better than a Nash equilibrium would. Alternatively, our agent might observe the other agents while interacting with them, and choose to change its strategy at runtime in order to improve its performance. However, deviating from a Nash equilibrium strategy also risks lower performance, if the agent's beliefs about the opponent are inaccurate. By learning about the other agents and using this knowledge, our agent can begin to trade off between minimizing risk and maximizing utility by using a **robust counter-strategy**. Fortunately this tradeoff is not necessarily linear, and it is possible to greatly improve an agent's expected performance in exchange for only a small penalty to its worst-case performance.

1.3 An Overview of This Thesis

In this paper-based thesis, we will explore the end-to-end task of creating strong game-playing agents through the Abstraction-Solving-Translation procedure. My contributions to this field form a cohesive body of work that has advanced the state-of-the-art in each of the steps listed above:

- Two new game solving algorithms, PCS-CFR and CFR-BR, that offer both efficiency and qualitative improvements over earlier algorithms.
- State-space abstraction techniques that create effective models of large games, and evaluation techniques for objectively measuring the quality of an abstraction.
- An unbiased and low-variance online evaluation technique for evaluating an agent's performance against competitors.
- An efficient worst-case analysis algorithm for precisely measuring an agent's suboptimality as compared to a Nash equilibrium.
- Online and offline techniques for modelling other agents, creating counter-strategies to use against them, and choosing which counter-strategy to use.

In the poker domain, my research has contributed to the first meaningful victory of a computer program over top human poker professionals [75], and to our solv-

ing of the game of heads-up limit Texas hold'em, the first human-scale imperfect information game to be essentially solved [17].

Each of these steps – abstraction and translation, game solving, evaluating in-game performance, evaluating worst-case performance, opponent modelling, and online adaptation – represents a significant problem area that has been the subject of research by myself, my colleagues at the University of Alberta, and by my fellow researchers around the world. Each chapter of this paper-based thesis will consist of one of my previously published research papers that explores one or more of these problem areas. The arc labels in Figure 1.2 indicate the chapters that discuss each task. My research papers presented in this thesis are:

- **Accelerating Best Response Calculation in Large Extensive Games** [53]. IJCAI, 2011. Presented in Chapter 2.
- **Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization** [50]. AAMAS, 2012. Presented in Chapter 3.
- **Finding Optimal Abstract Strategies in Extensive-Form Games** [49]. AAAI, 2012. Presented in Chapter 4.
- **Evaluating State-Space Abstractions in Extensive-Form Games** [52]. AAMAS, 2013. Presented in Chapter 5.
- **Data Biased Robust Counter Strategies** [51]. AISTATS, 2009. Presented in Chapter 6.
- **Strategy Evaluation in Extensive Games with Importance Sampling** [19]. ICML, 2008. Presented in Chapter 7.
- **Heads-up Limit Hold'em Poker is Solved** [17]. Science, 2015. Presented in Chapter 9.

In addition to these published works, Chapter 8 will present an analysis of the 2007 and 2008 Man-vs-Machine Poker Championships. In these events our poker-playing agent "Polaris" competed against human poker pros, narrowly losing in 2007 and narrowly winning in 2008. The victory in 2008 marked the first time that a poker-playing computer defeated human professionals in a meaningful competition. While the raw results of the competitions were not statistically significant, the postgame analysis technique developed in Chapter 7 allows us to derive an unbiased low-variance estimate of Polaris' performance. For the first time, we will present an analysis of the 2007 and 2008 competitions using this technique, and demonstrate that Polaris earned its victory.

Throughout this thesis, the ideas presented will be validated using the game of poker as a testbed domain. Poker is the canonical game of imperfect information, and is famous for its themes of bluffing, deception, and psychological trickery and insight, which are not typically considered "computer-like" qualities. The game is

appealing to us for the same reasons that it was appealing to von Neumann: by introducing imperfect information, stochastic outcomes, one or more adversaries of varying skill, and an emphasis on maximizing winnings, poker represents steps in several dimensions along the gradient towards real-life domains. To validate the progress of our research in the poker domain, we will use many methods: comparisons against other computer agents in the Annual Computer Poker Competition, comparisons against human experts in the 2007 and 2008 Man-vs-Machine Poker Championships, objective measurements of our agents' suboptimality as compared to a Nash equilibrium, and finally, our ability to solve an ever-larger series of games, culminating in our solution to heads-up limit Texas hold'em poker.

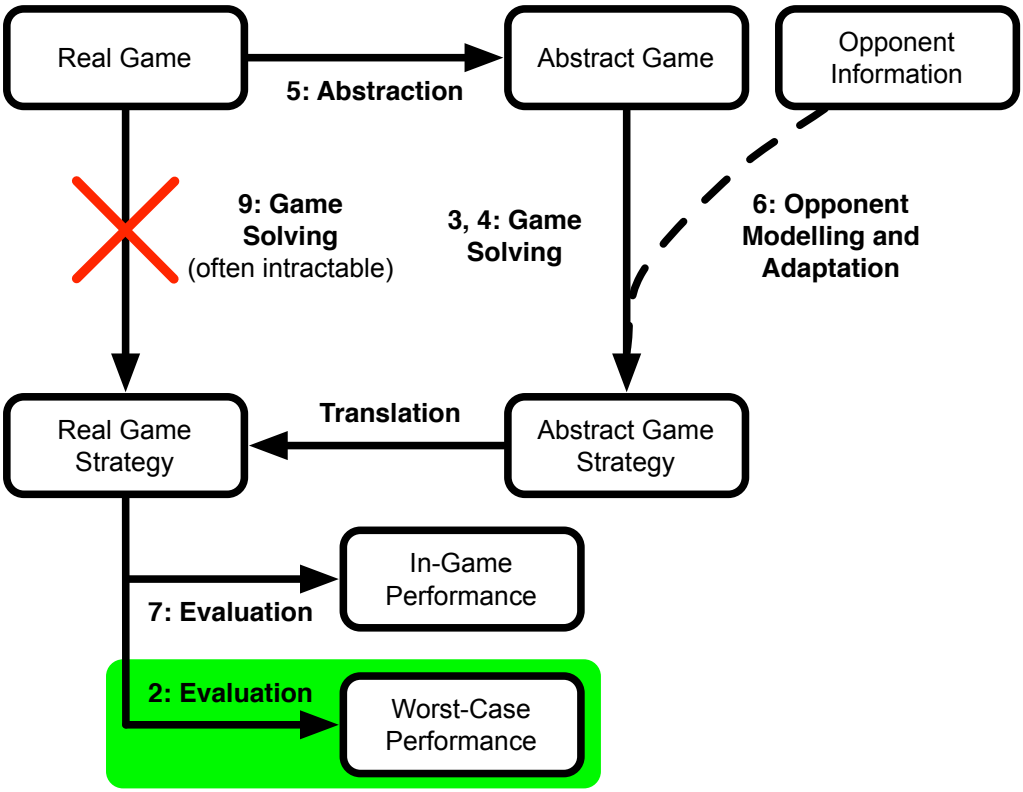
Chapter 2

Worst-Case Evaluation

Accelerating Best Response Calculation in Large Extensive Games

Michael Johanson, Kevin Waugh, Michael Bowling, Martin Zinkevich

Presented at IJCAI 2011 [53]



When we use the Abstraction-Solving-Translation method to approximate an optimal Nash equilibrium strategy, the natural question to ask is: how close of an approximation is it? Errors can be introduced into a strategy from many sources, such as the lossy abstraction and translation methods that make the computation tractable, the lack of theoretical guarantees on real-game performance when solving an abstract game, the use of a game solving algorithm that is stopped before an optimal strategy is reached, software errors, and so on. Calculating the suboptimality of a strategy allows us to measure these effects, and by measuring suboptimality over time, we can evaluate the progress of our research.

Exploitability is the metric by which we measure this approximation. In a two-player zero-sum game, Nash equilibria are typified by being unexploitable: they can do no worse than tie, on expectation, against any adversary. Given an arbitrary strategy, we can measure its suboptimality by measuring how much it will lose, on expectation, against a worst-case "best response" opponent strategy. This evaluation method lets us measure suboptimality even when a Nash equilibrium cannot itself be tractably computed.[1]

However, computing a best response to an arbitrary strategy can be a challenging computational task in nontrivial games. The smallest human-scale poker game of heads-up limit Texas hold'em was the inaugural game used in the Annual Computer Poker Competition, and has been the primary research domain used by researchers until recently. However, it was only in 2011 that my colleagues and I developed the first algorithm capable of tractably computing best responses for arbitrary strategies in large extensive form games such as HULHE, called "Accelerated Best Response". This came long after the first game theoretic agent PsOpti was developed in 2003 [11], the first CFR strategies were computed in 2007 [110], or Polaris' victory over human professionals in 2008 [75]. While all of these earlier agents were intended to approximate Nash equilibria, and the later strategies defeated earlier ones in one-on-one play, their suboptimality was unknown until our technique was developed and it was unclear whether exploitability was actually decreasing over time.

With the development of Accelerated Best Response, we were able for the first time to evaluate the progress of our research over the previous five years. Additionally, by collaborating with our colleagues in the Annual Computer Poker Competition, we evaluated many of the 2010 ACPC agents to evaluate the community as a whole. Our experiments demonstrated that consistent progress towards approximating a Nash equilibrium had in fact been made, although even our human professional calibre agent Polaris was shockingly exploitable by human standards.

After its development, the algorithm also guided the next five years of research up to the present as it allowed us to evaluate abstraction techniques, measure the convergence rate of new game solving algorithms, and investigate a new form of "abstraction pathology" in which convergence in an abstract game can produce sub-

[1] If we already know a Nash equilibrium strategy, then we might use other metrics that more directly compare the strategies, such as by comparing the action probabilities at each information set. However, this approach is fraught with difficulties: what if multiple equilibria exist, how similar are strategies where one takes an action that the other never does, and so on.

optimal behaviour in the real game. The algorithm also contributed vital insights towards the development of new game solving techniques. The key contribution of the algorithm is a traversal of the game's public states, during which we consider the large vectors of private states that each player can be in. This public tree traversal was used to form the PCS-CFR [50], CFR-BR [49], CFR-D [21], and CFR+ [95] game solving algorithms, which offer improved efficiency and qualitative differences over the base CFR game solving algorithm. PCS-CFR, CFR-BR, and CFR+ will be presented in Chapters 3, 4, and 9 respectively.

Author's contributions. Zinkevich and Waugh are responsible for the fast vector-based terminal node evaluation that reduces the $O(n^2)$ operation to $O(n)$. I am responsible for the application of this trick to our large extensive form games. I designed and implemented the algorithm, and collected and interpreted all of the empirical results. Computing the exploitability of the ACPC agents required the assistance of the agents' programmers (Andersen, Byrnes, Ciucu, Ganzfried, and Lin), who worked with me to run the experiment. The paper was principally written by myself and Bowling, with assistance from Waugh and Zinkevich.

Accelerating Best Response Calculation in Large Extensive Games[2][3]

Michael Johanson (johanson@ualberta.ca)
Kevin Waugh (waugh@cs.cmu.edu)
Michael Bowling (bowling@ualberta.ca)
Martin Zinkevich (maz@yahoo-inc.com)

Abstract:
One fundamental evaluation criteria of an AI technique is its performance in the worst-case. For static strategies in extensive games, this can be computed using a best response computation. Conventionally, this requires a full game tree traversal. For very large games, such as poker, that traversal is infeasible to perform on modern hardware. In this paper, we detail a general technique for best response computations that can often avoid a full game tree traversal. Additionally, our method is specifically well-suited for parallel environments. We apply this approach to computing the worst-case performance of a number of strategies in heads-up limit Texas hold'em, which, prior to this work, was not possible. We explore these results thoroughly as they provide insight into the effects of abstraction on worst-case performance in large imperfect information games. This is a topic that has received much attention, but could not previously be examined outside of toy domains.

2.1 Introduction

Extensive games are a powerful framework for modelling sequential stochastic multi-agent interactions. They encompass both perfect and imperfect information games allowing work on extensive games to impact a diverse array of applications.

To evaluate an extensive game strategy, there are typically two options. The first option is to acquire a number of strategies and use an appropriately structured tournament. This is, by far, the most common option in the AI community, and is used by the Annual Computer Poker Competition, the Trading Agent Competition, RoboCup, the General Game Playing competition, SAT and Planning competitions, and so on. While the tournament structure will ultimately declare a winner, it is not always clear how to interpret the results. That is, which strategy is indeed the best when the results are noisy, have intransitivities, or there are multiple evaluation criteria?

The second option is to compute or bound the worst-case performance of a strategy. Good worst-case performance suggests a strategy is robust to the choices of the other players. In zero-sum games, the worst-case performance takes on additional

[2]The paper presented in this chapter originally appeared at the *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*. Copyright 2011 International Joint Conferences on Artificial Intelligence. M. Johanson, K. Waugh, M. Bowling, and M. Zinkevich. Accelerating best response calculation in large extensive games. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 258–265, 2011.

[3]The Appendix has additional results related to this paper.

importance, as it is intimately tied to the Nash equilibrium concept. A strategy's worst-case performance can be computed using a best response calculation, a fundamental computation in game theory. Conventionally, the best response computation begins by examining each state once to compute the value of every outcome, followed by a pass over the strategy space to determine the optimal counter-strategy. For many games, this is infeasible with modern hardware, despite requiring only a single traversal of the game. For example, two-player limit Texas hold'em has 9.17×10^{17} game states, which would require ten years to examine even if one could process three billion states per second.

Since best response computation has been thought to be intractable for two-player limit Texas hold'em, evaluation has focused on competitions instead. The Annual Computer Poker Competition holds an instant runoff tournament for variants of Texas hold'em poker. Historically, top entrants to this tournament have aimed to approximate Nash equilibrium strategies using increasingly finer abstractions and efficient equilibrium-finding techniques. More accurate solutions to ever finer abstractions were thought to improve worst-case performance and result in stronger tournament performance. While tournament results suggested this was happening, recent work shows that finer abstractions give no guarantee of improving worst-case performance [103]. This finding was demonstrated on a toy domain where best-response computations were feasible. To date, no one has measured the worst-case performance of a single non-trivial strategy in any of the competition events.

In this paper, we describe general techniques for accelerating best response calculations. The method uses the structure of information and utilities to avoid a full game tree traversal, while also being well-suited to parallel computation. As a result we are able for the first time to compute the worst-case performance of non-trivial strategies in two-player limit Texas hold'em. After introducing these innovations, we use our technique to empirically answer a number of open questions related to abstraction and equilibrium approximation. We show that in practice finer poker abstractions do produce better equilibrium approximations, but better worst-case performance does not always result in better performance in a tournament. These conclusions are drawn from evaluating the worst-case performance of over three dozen strategies involving ten total CPU years of computation.

2.2 Background

We begin with a brief description of an extensive game; a formal definition can be found in [76, Ch. 11]. An extensive game is a natural model of interaction between players in an environment. A **history** $h \in H$ is the sequence of actions taken by all of the players, including the **chance player** whose actions represent random events such as card deals or dice rolls. A **player function** $P(h)$ determines which player is next to act. Taking an action a in a history h produces a new history ha. A subset of all histories are **terminal histories**. At a terminal history z, the game is over and the players are assigned utilities according to a **utility function**, where $u_i(z)$

is the utility for player i. If there are two players and the utilities sum to 0 (so $u_1(z) = -u_2(z)$ for all z), then we say the game is **zero-sum**. An extensive game can be intuitively thought of as a game tree, where each history is a game state and actions by the players cause transitions to new histories.

In games of **imperfect information**, some of the actions by the players, or chance, may not be fully revealed. One example is the game of poker, in which chance deals private cards face-down to each player, and these cards determine the utilities at the end of the game. To represent such cases, we use **information sets** to group indistinguishable histories. If player 1 cannot see player 2's cards, for example, then all of the histories that are differentiated only by player 2's cards are in the same information set for player 1. When a player is to choose an action, their choice only depends on the information set, not the underlying history.

A **strategy** for a player i, $\sigma_i \in \Sigma_i$, is a function that assigns a probability distribution over the actions to each information set I. When player i must select an action at I, they sample an action from the probability distribution $\sigma_i(I)$. Note that in games such as poker where the agents alternate positions after each game, an agent will have one strategy to use in each position. A **player** has a specified position in the game, while an **agent** has a strategy for every position in the game.

A **strategy profile**, $\sigma \in \Sigma$, consists of a strategy for each player. We use σ_{-i} to refer to all of the strategies in σ except for σ_i. As a strategy profile defines the probability distribution over actions for all of the non-chance players, it is sufficient to determine the expected utility for each player, denoted $u_i(\sigma)$. Similarly, we define $u_i(\sigma_1, \sigma_2) = u_i(\sigma_1 \cup \sigma_2)$.

A **best response** is the optimal strategy for player i to use against the opponent profile σ_{-i}. It is defined as

$$b_i(\sigma_{-i}) = \underset{\sigma_i' \in \Sigma_i}{\operatorname{argmax}}\, u_i(\sigma_{-i}, \sigma_i'). \tag{2.1}$$

The value of the best response, $u_i(b_i(\sigma_{-i}), \sigma_{-i})$, is how much utility the best response will receive on expectation. In two player games, this value is also useful as a worst-case evaluation of the strategy σ_i, as $u_i(\sigma_i, b_{-i}(\sigma_i))$ is a lower bound on player i's utility on expectation.

Two-player zero-sum games have a **game value**, v_i, that is the lower bound on the utility of an optimal player in position i. In this case, we use the term **exploitability** to refer to the difference

$$\varepsilon_i(\sigma_i) = v_i - u_i(\sigma_i, b_{-i}(\sigma_i)). \tag{2.2}$$

The exploitability of a strategy is thus how much additional utility is lost to a worst-case adversary by playing σ_i, instead of a strategy that achieves the game's value. A strategy is **unexploitable**, or optimal, if this difference is zero.

In large two player zero-sum games such as poker, the value of the game is unknown and is intractable to compute; however, if the players alternate positions, then the value of a pair of games is zero. If an agent plays according to profile σ then its exploitability is

$$\varepsilon(\sigma) = \frac{u_2(\sigma_1, b_2(\sigma_1)) + u_1(b_1(\sigma_2), \sigma_2)}{2}. \tag{2.3}$$

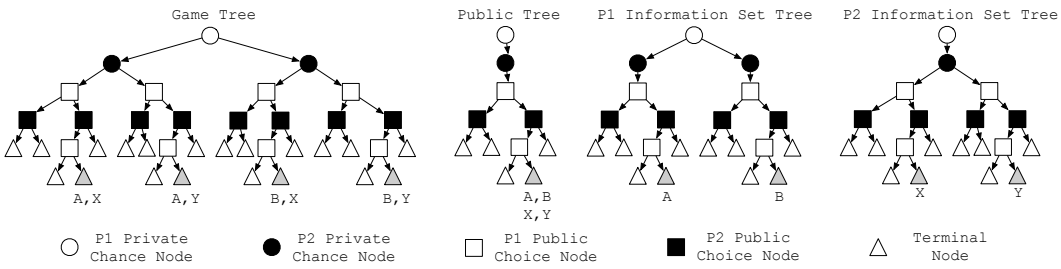


Figure 2.1: Four trees for the game of 1-Card Poker.

A Nash equilibrium is a strategy profile σ that has the property that no player can benefit by unilaterally deviating.

Definition 1 *σ is a **Nash Equilibrium** if*

$$u_i(\sigma) \geq u_i(\sigma_{-i} \cup \sigma'_i), \forall i \in N, \forall \sigma'_i \in \Sigma_i. \tag{2.4}$$

In two player zero-sum games, a Nash equilibrium is unexploitable: against any opponent the strategy will win no less than the value of the game. However, in large games it may be intractable to compute such a strategy. In such cases, an aligned goal is to attempt to approximate an equilibrium with a low exploitability strategy, thus bounding its worst-case performance. Various abstraction and equilibrium computation methods have been applied to making what were hoped to be good equilibrium approximations. While many of these have been applied to building strategies for two-player limit Texas hold'em, it has been previously infeasible to evaluate their worst-case performance in such a large domain.

2.3 Conventional Best Response

Conventionally, a best response can be easily computed through a recursive tree walk that visits each game state once. To illustrate this algorithm, we will refer to Figure 2.1, which presents four views of the simple game of 1-Card Poker. Consider the diagram labelled "Game Tree", which represents the exact state of the game. The game begins at the root, where the white circle nodes represent chance privately giving one card to Player 1. The children, black circles, represent chance privately giving one card to Player 2. Descending through the tree, the white and black squares represent players 1 and 2 respectively making public betting actions, before arriving at the terminal nodes, represented by triangles.

Since there is private information in this game, each player has a different view of the game, which is represented by the "P1 Information Set Tree" and "P2 Information Set Tree" diagrams. In these diagrams, the private information provided to the opponent by chance is unknown, and so the black and white circles for the opponent have only one child. Each node in these trees represents a set of game states that the player cannot distinguish. For example, if Player 1 reaches the terminal

node labelled 'A' in their information set tree, they cannot determine whether they are in the game tree node labelled 'A,X' or 'A,Y', as these nodes are distinguished only by the opponent's private information, while all of the public actions leading to these nodes are identical.

Consider computing a best response from Player 2's perspective. We will do a recursive tree walk over their information set tree. At a terminal node, such as the one labelled 'X', we must consider all of the game states that the game could actually be in: 'A,X' or 'B,X'. Specifically, we need to know the probability that the opponent would reach the nodes 'A' and 'B' in their tree, according to their strategy being used at their earlier decision points and chance's strategy; we will call this vector of probabilities the **reach probabilities**. Given these probabilities, we can compute the unnormalized value for us reaching 'X' to be the sum over the indistinguishable game states of each game state's utility times that game state's reach probability. We then return this value during our tree walk. Recursing back through our choice nodes, the black squares, we will pick the highest valued action to create our best response strategy, and return the value of this action. At opponent choice nodes and chance nodes, the white squares and circles, we simply return the sum of the child values. When we return to the root, the returned value is the value of the best response to the opponent's strategy. Performing this computation from each position gives us both best response values, and thus by Equation 2.3, the exploitability of the strategy.

Note that there is an obvious enhancement to this algorithm. Instead of recomputing the opponent's vector of reach probabilities at each terminal node, during our recursive tree walk we can pass forward a vector containing the product of probabilities of the earlier actions in the tree. This allows us to query the opponent's strategy once for each of its earlier actions and reuse the strategy's action probability at all of its descendent terminal nodes. In large domains, the opponent's strategy might be many gigabytes and be stored on disk or otherwise have a nontrivial cost to do such queries, and it is important to take advantage of opportunities to cache and reuse these computed values. Thus, our recursive tree walk will pass forward a vector of reach probabilities for the states in an information set, and return the value for reaching an information set.

2.4 Accelerated Best Response

The conventional best-response computation visits each game state exactly once, and is thus seemingly efficient. However, in large games such as Texas hold'em poker, with 10^{18} states, having to visit each state once makes the computation intractable. In this section, we will show four ways that this conventional algorithm can be accelerated: (1) traversing a different type of tree, which allows more opportunities for caching and reusing information; (2) using properties of the game's utilities to efficiently evaluate terminal nodes of the public tree; (3) use game-specific isomorphisms to further reduce the size of the expanded tree; (4) solving independent sections of this new tree in parallel.

Public State Tree. We begin by presenting the heart of the accelerated best response algorithm.

Definition 2 (Public State) *We call a partition of the histories, $\mathcal{P}$, a **public partition** and $P \in \mathcal{P}$ a **public state** if*

- *no two histories in the same information set are in different public states (i.e., if information is public, all players know it)*
- *two histories in different public states have no descendants in the same public state (i.e., it forms a tree), and*
- *no public state contains both terminal and non-terminal histories (we call a public state with terminal histories a **terminal public state**).*

Informally, a public state is defined by all of the information that both players know, or equivalently, what the game looks like to an observer that knows no private information. Like the histories, it forms a tree that we can traverse. Though it is not provided by an extensive game's description, it is trivial to come up with a valid public information partition, and many games have a natural notion of public information. Following our earlier example of 1-Card Poker, Figure 2.1 shows an example of the public tree, beside the much larger game tree.

In our earlier example illustrating the conventional algorithm, we used Figure 2.1 to describe that when we are at terminal node X, we do not know if the opponent is at node A or B, and so we must compute the reach probabilities for the opponent and chance reaching each of these states. However, there is an important opportunity to reuse these probabilities; just as we cannot distinguish between their private states, the opponent cannot distinguish if we are at node X or node Y, and so the reach probabilities are identical when we are in either state.

The public tree provides a structured way to reuse these computed probabilities. Every node in the public tree represents a set of game states that cannot be distinguished by an outside observer, and also partitions the information sets for the players. Instead of finding a best response by walking over the information set tree, as in the conventional algorithm, we will instead recursively walk the much smaller public tree. When we reach a terminal node such as the one labelled "A,B,X,Y" in Figure 2.1, we know that player 1 could be in nodes A or B as viewed by player 2, and that player 2 could be in nodes X or Y as viewed by player 1. From player 2's perspective, we can calculate the vector of player 1's reach probabilities for A and B once, and reuse these probabilities when computing the value for both X and Y. Instead of passing forward a vector of reach probabilities and returning a single value for the one information set being considered, as we do in the conventional algorithm, we will instead pass forward a vector of reach probabilities and return a vector of values, one for each of our information sets in the public state. At our decision nodes, we will pick the best action for each information set by recursing to get a vector of values for each action, and return a vector where each entry is the max of that entry across the action value vectors. At opponent decision nodes and

chance nodes, we will recurse to find the child value vectors, and return the vector sum of these vectors. At terminal nodes, the value for each information set can be found by evaluating each one at a time, as in the conventional algorithm, although later we will present a faster technique that produces the same values.

Thus, the public tree walk performs exactly the same computations as the conventional algorithm, and only the order of these computations has changed so that we can more efficiently reuse the queries to the opponent's strategy. In a game like Texas hold'em where each player has up to 1326 information sets in each public state, this allows us to avoid 1325 unnecessary strategy queries. As previously mentioned, if querying the opponent's strategy is nontrivial and forms a bottleneck in the computation, the speed advantage is substantial, as it may do as little as $\frac{1}{1326}$ times as much work. In practice, we estimate that this change may have resulted in a 110 times speedup.

Efficient Terminal Node Evaluation. The second way to accelerate the calculation involves improving the terminal public state utility calculation by exploiting domain specific properties of the game. When reaching terminal public states during the traversal, we have a vector of probabilities for the opponent reaching that public state with each of their information sets, and need to compute the value of each of our information sets. A naive approach is to consider each pair of information sets in the public state. This is an $O(n^2)$ computation, assuming n information sets per player. However, games of interest typically have structure in their payoffs. Very often this structure can allow an $O(n)$ computation at the terminal nodes, particularly when the distribution over the players' information sets are (nearly) independent.

Example 1. Suppose the players' information sets are factorable and only some of the factors affect the utilities at any particular public state. For example, in a negotiation game, a player may have information that affects the utility of many different negotiated outcomes, but only the information associated with the actual negotiated outcome affects that public state's utility. If the number of *relevant information sets* is only $O(\sqrt{n})$ and the information sets are independently distributed, then the $O(n^2)$ computation can be done in only $O(n)$ time.

Example 2. Suppose the players' information sets make independent and additive contributions to the best-response player's utility. For example, consider a game where goods are being distributed and the players have independent estimates for a good's true value. If the player's utility is the true value of the acquired goods, then each player's estimate is making an independent contribution. In such a situation, the expected opponent's contribution can be first computed independently of the best-response player's information set, allowing the whole utility calculation to be completed in $O(n)$ time.

Example 3. Suppose we can sort each players' information sets by "rank", and the utility only depends upon the relative ordering of the players' ranks. This is exactly the situation that occurs in poker. For the moment, let us assume the distribution of the players' ranks are independent. In this case, evaluating each of our

information sets requires only $O(n)$ work. We know that our weakest information set will be weaker than some of the opponent's hands, equal to some, and better than some. We keep indices into the opponent's ordered list of information sets to mark where these changes occur. To evaluate our information set, we only need to know the total probability of the opponent's information sets in these three sections. After we evaluate one of our information sets and move to a stronger one, we just adjust these two indices up one step in rank.

This approach can be used in cases where the players' hidden information is independent. However, even if the information sets are dependent, as in poker where one player holding a card excludes other players from holding it, we may still be able to do an $O(n)$ evaluation. We will use the game of Texas hold'em as an example. In this game, a 52-card deck is used, five cards are revealed, and each player holds two cards. We proceed as before, and evaluate each of our (47 choose 2) possible hands, considering all (47 choose 2) hands for the opponent. However, some of the opponent's hands are not possible, as their cards overlap with ours. Using the inclusion-exclusion principle, when computing the total probability of hands better and worse than ours, we subtract the total probability of opponent hands that include either of our cards. The opponent hand that uses both of our cards has then been incorrectly subtracted twice, so we correct by adding its probability back again. This $O(n)$ procedure results in exactly the same value as the straightforward $O(n^2)$ evaluation. In games such as poker, the ordering of the information sets may not be known until the final action by the chance player is revealed. In such cases, the information sets must be sorted after this occurs, resulting in an $O(n \log n)$ sorting procedure followed by the $O(n)$ terminal node evaluation. However, the $O(n \log n)$ sorting cost can be paid once and amortized over all of the terminal node evaluations following the final chance event, reducing its cost in practice. In Texas hold'em, this $O(n \log n)$ evaluation runs 7.7 times faster than the straightforward $O(n^2)$ evaluation.

Game-Specific Isomorphisms. The third way to accelerate the calculation depends on leveraging known properties of a strategy. In some games, there may be actions or events that are strategically equivalent. This is true in many card games, where the rank of a card, but not its suit, indicates strength. For example, a $2\heartsuit$ may be equivalent to a $2\spadesuit$, at least until additional cards are revealed; if the chance player later reveals a $3\spadesuit$, $2\spadesuit$ may then be stronger than a $2\heartsuit$. We call such sets of equivalent chance actions **isomorphic**, and choose one arbitrarily to be **canonical**. If the domain has this property and if the strategy being evaluated is the same for every member of each set of isomorphic histories, then the size of the public state tree can be greatly reduced by only considering canonical actions. On returning through a chance node during the tree walk, the utility of a canonical action must be weighted by the number of isomorphic states it represents. In Texas hold'em, this reduction results in a public state tree 21.5 times smaller than the full game.

Parallel Computation. The fourth and final way in which we accelerate the calculation is to choose subtrees of the public tree that can be solved independently. We rely on the fact that the graph of public states forms a tree, as required by Definition 2, and computing a value vector at a public state requires only the vector of probabilities for the opponent to reach this public state and computations on the subtree of public states descendent from it. Given this, any two public states where one is not descendent from the other will share no descendents and thus have no computations in common, and so can be solved in parallel. For example, when reaching a public chance event during a public tree walk, all of the children could be solved in parallel and the value vector returned as normal once each subtree computation has finished.

In Texas hold'em poker, one natural choice of a set of independent subgames to solve in parallel is at the start of the second round, called the "flop". There are 7 nonterminal action sequences in the first round and 1755 canonical public chance events at the start of the flop, resulting in 12,285 independent subgames for each position, and 24,570 subgames total. Using the accelerated best response technique described above, each subgame requires approximately 4.5 minutes on average to solve, resulting in a 76 CPU-day sequential computation. Since these subgames are independent, a linear speedup can be achieved by using 72 processors to solve the set of subgames in just over a day. When all of the subgames are complete, walking the small tree from the root to the start of the subgames requires less than a minute to complete.

The four methods described above provide orthogonal speed enhancements over the conventional best response algorithm. By combining them, we can now compute the value of a best response in just over a day, in a domain where the computation was previously considered intractable.

2.5 Application to Texas Hold'em Poker

Our new ability to calculate the exploitability of strategies in large extensive games allows us to answer open questions about abstraction and approximate equilibria which have been raised by the computer poker community. The Annual Computer Poker Competition, which was started in 2006, has popularized poker as a challenging testbed for artificial intelligence research. Annually, over two dozen poker-playing programs are submitted to the competition. Although many approaches are used, the most popular technique is to approximate an unexploitable Nash equilibrium strategy. However, the smallest and most popular variant of poker in the competition, heads-up Limit Texas hold'em, has $9.17 * 10^{17}$ game states, rendering the computation of an exact solution intractable. Instead, a compromise is made that is common to many large domains in artificial intelligence: a smaller abstract game is constructed and solved, and the resulting abstract strategy is used in the real game.

With this approach, the competitors have made substantial progress in discovering more efficient game solving techniques, such as Counterfactual Regret Min-

imization [110] and the Excessive Gap Technique [41]. More efficient algorithms and more powerful hardware has allowed for larger, finer-grained abstractions to be solved. These new abstractions have involved imperfect recall and a focus on public information [104], and k-means-clustering-based approaches that better model the real game [39]. This line of work has been motivated by an intuition that larger, finer-grained abstractions will produce less exploitable strategies.

Unfortunately, recent work has shown this intuition to be unreliable. In a toy poker game, counterexamples were found where refining an abstraction to a finer-grained one produced strategies that were dramatically more exploitable [103]. These **abstraction pathologies**, if present in Texas hold'em, could result in highly exploitable agents. As the best response calculation was until now intractable, their possible effect has been unknown.

In the next section, we will present results from our best response technique in two-player limit Texas hold'em. In particular, we aim to answer three key questions from the computer poker community. First, how exploitable are the competition's approximations to equilibrium strategies? Second, is progress being made towards the goal of producing an unexploitable strategy? Third, do abstraction pathologies play a large role in Texas hold'em? In addition, we will raise new issues related to abstraction and equilibria that these experiments have revealed.

2.6 Results in the Texas Hold'em Domain

We will begin by presenting the exploitability of several trivial Texas hold'em agents, shown in Table 2.1, to give context to later results. All of our results are presented in milli-big-blinds per game (mbb/g) won by the best response (or **milliblinds**), where a milli-big-blind is 0.001 big blinds, the unit of the largest ante in Texas hold'em.[4] Note that the exploitability values presented are precise to within floating-point inaccuracies. The "Always Fold" agent always chooses the fold action to surrender the game. Thus, its exploitability is trivially calculable to be 750 mbb/g without our analysis. However, the exploitability of the "Always-Call", "Always-Raise", and "50% Call 50% Raise" agents are not trivially computable. The exploitability of "Always-Raise" has been independently computed [71] and matches our result after changing units, but to our knowledge, our analysis is the first for "Always-Call" and "50% Call 50% Raise". For comparison, a rule-of-thumb used by human poker professionals is that a strong player should aim to win at least 50 mbb/g.

[4] In older papers [110; 54], the unit "milli-small-bets per game (mb/g)" was used. In heads-up limit Texas hold'em, a small bet is equal in size to a big blind, and so milli-small-bets and milli-big-blinds are equivalent. This paper adopted the milli-blind unit (mb/g) for consistency with other games (such as no-limit poker) where blinds are used but bet sizes are not specified. In later papers, the milli-blind unit (mb/g) was disambiguated to milli-big-blind (mbb/g). In this chapter, we have adjusted the original paper's mb/g units to mbb/g, for consistency with the other papers presented in this thesis.

Agent Name	Exploitability (mbb/g)
Always-Fold	750
Always-Call	1163.48
Always-Raise	3697.69
50% Call, 50% Raise	2406.55

Table 2.1: Exploitability of four trivial Texas Hold'em Poker agents. Exploitability is measured in milli-big-blinds per game lost to a best response.

Name	vs (4)	Exploitability (mbb/g)
(1) Hyperborean.IRO	-3 $\pm$ 2	135.427
(2) Hyperborean.TBR	-1 $\pm$ 4	141.363
(3) GGValuta	-7 $\pm$ 2	237.330
(4) Rockhopper	0	300.032
(5) GS6.IRO	-37 $\pm$ 6	318.465
(6) PULPO	-9 $\pm$ 2	399.387
(7) Littlerock	-77 $\pm$ 5	421.850

Table 2.2: Agents from the 2010 Computer Poker Competition. The "vs (4)" column shows the performance and 95% confidence interval against the top-ranked agent in the competition. The "Exploitability" column shows the expected loss to a best response in milli-big-blinds per game. Extended results are available in the Appendix as Tables 11.1 and 11.2.

Computer Poker Competition Results. The Annual Computer Poker Competition is a driving force behind recent research into equilibrium-finding and abstraction techniques. However, due to the computational complexity of computing a best response using conventional techniques, the worst-case performance of the agents was unknown. With the cooperation of the agents' authors, we have used our technique to calculate the exact exploitability of some of the agents that competed in the 2010 ACPC. These results are presented in Table 2.2. A complete table of the agents' relative performance can be found on the competition's website [44].

From Table 2.2, we see that there is a wide range in exploitability between these agents, even though the first five appear to be similar from the tournament results. This suggests that while this one-on-one performance gives us a bound on the exploitability of a strategy, it does not indicate how far away from optimal a strategy is. Note that the PULPO strategy is not explicitly attempting to approximate an unexploitable strategy as the other agents are; it uses a pure strategy that gives up exploitability in return for better one-on-one performance against weak opponents. In fact, PULPO was the winner of the 2010 Bankroll event, in which agents attempt to maximize their utility against the other competitors. If the goal is to create an agent that performs well against other agents, having the lowest exploitability is not sufficient to distinguish a strategy from its competitors, or to win the competition.

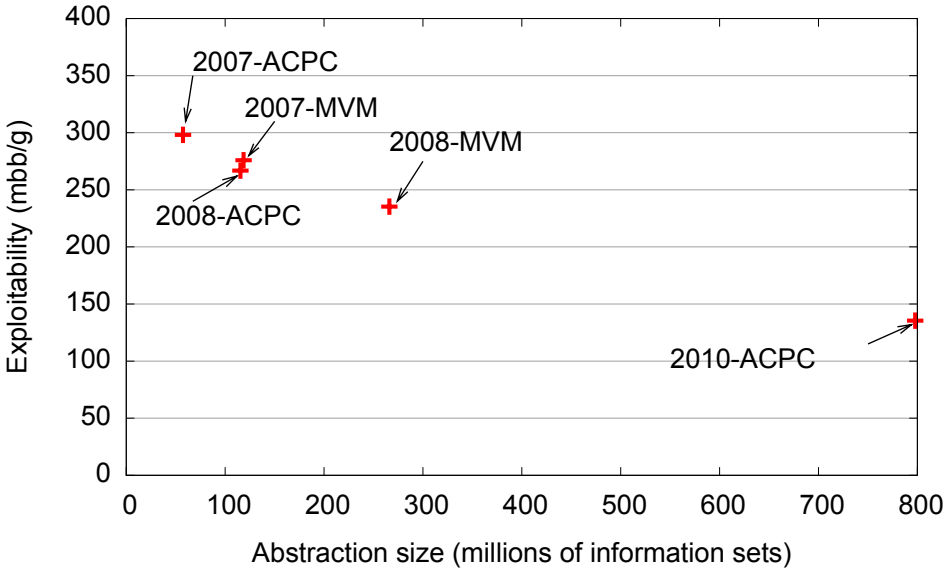


Figure 2.2: Exploitability of the University of Alberta's competition strategies over a four year span.

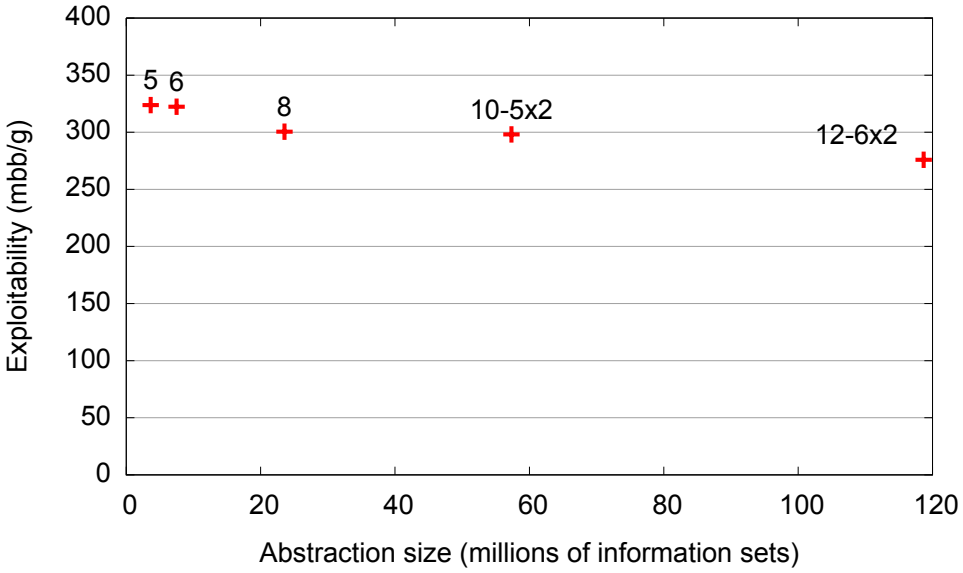


Figure 2.3: Abstraction size versus exploitability.

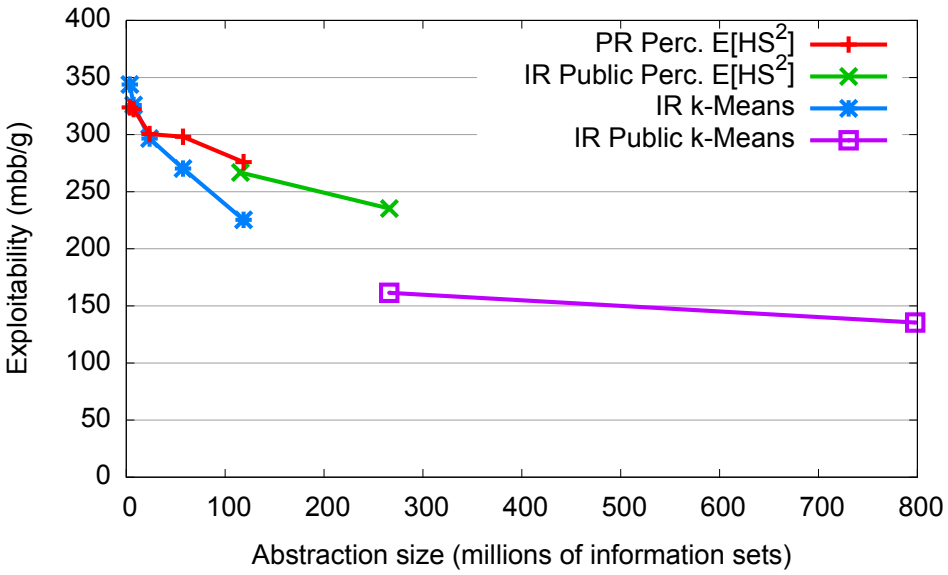


Figure 2.4: Four different abstraction techniques as the size of the abstraction varies.

Abstraction Pathologies. As we mentioned in Section 2.5, toy domains have shown that increasing one's abstraction size (even strictly refining an abstraction) does not guarantee an improvement, or even no change, in worst-case performance. Instead, examples in these domains show that even strict refinements can result in more exploitable strategies. With the accelerated best response technique, we can now for the first time explore this phenomenon in a large domain. In Figure 2.2, we present an analysis of the University of Alberta Computer Poker Research Group's entries into the Annual Computer Poker Competition and two Man-vs-Machine competitions over a period of four years. Over time, improvements in the implementation of our game solving algorithm [110] and access to new hardware have allowed for larger abstractions to be created and solved. This increased abstraction size has also allowed for more complex abstraction techniques that use new domain features. In turn, this has led to a consistent decrease in the exploitability of the strategies.

In Figure 2.3, we consider increasing sizes of abstractions generated by one particular abstraction methodology (Percentile Hand Strength) using the Counterfactual Regret Minimization algorithm [110] to solve the abstraction. At each chance node, the possible outcomes are ranked according to the Expected Hand Strength Squared (E[HS^2]) metric and divided equally into a number of buckets. For the larger 10-5x2 and 12-6x2 bucket abstractions, the hands were first divided into 5 and 6 buckets respectively according to the E[HS^2] metric then each was further split into 2 buckets according to the E[HS] metric. This means that we have two examples of strict refinement as described by [103]: 10-5x2 is a strict refinement of 5, and 12-6x2 is a strict refinement of 6. In the chart, we see that increasing the size of the abstraction provides a consistent, albeit small improvement.

Name	Abs. Size	Tilt %	Exploitability (mbb/g)
Pink	266m	0,0,0,0	235.294
Orange	266m	7,0,0,7	227.457
Peach	266m	0,0,0,7	228.325
Red	115m	0,-7,0,0	257.231
Green	115m	0,-7,0,-7	263.702
Reference	115m	0,0,0,0	266.797

Table 2.3: Analysis of the 5 component strategies in the "Polaris" agent that competed in the 2008 Man Machine Poker Championship. "Tilt %" shows the percent added to a player's payoff when they win a showdown, lose a showdown, fold, or win due to their opponent folding. "Exploitability" is calculated in the unmodified game.

Finally, in Figure 2.4, we show the results of varying the abstraction size for four different abstraction techniques. The "PR Perc. E[HS^2]" abstraction technique has perfect recall and uses the Percentile Hand Strength technique as described in [110]. The "IR Public Perc. E[HS^2]" abstraction technique uses imperfect recall and public information as described in [104]. The two "k-Means" abstraction families also use imperfect recall and the same public buckets, and also use a k-means-clustering technique based on a player's hand's expected value and its potential to improve. In all four cases, increasing the abstraction size results in lower exploitability.

From Figures 2.2, 2.3 and 2.4, it appears that the abstraction pathologies encountered in small games do not appear to be common in the types and sizes of abstractions used in limit Texas hold'em. In these experiments, using one abstraction technique and solving increasingly larger games results in consistent decreases in exploitability. While diminishing returns affect the result, this decline appears predictable.

Tilting the payoffs. While Nash equilibrium approximations are robust against any opponent, they do not exploit all of the mistakes made by their opponents. Human domain knowledge in poker suggests that an "aggressive" strategy that chooses betting options more frequently, while making an exploitable mistake, may perform better against weak opponents. In 2008's Man-vs-Machine competition, the Polaris agent included off-equilibrium aggressive strategies that were created by running the counterfactual regret minimization algorithm on a variety of non-zero-sum games that asymmetrically increased the payoff for the winner or decreased the penalty for the loser. We refer to such slight modifications as **tilting** the game, and the resulting strategy as a **tilt**. Table 2.3 shows the five colour-named component strategies used in Polaris, along with the percent modification to the payoffs for when a player wins a showdown, loses a showdown, loses by folding, and wins by the opponent folding. Thus, the "Orange" agent believes it gets 7% more whenever it wins, and pays the normal penalty when it loses. "Pink" was an unmodified equilibrium; "Red" and "Green" used a smaller abstraction, and so an equilibrium

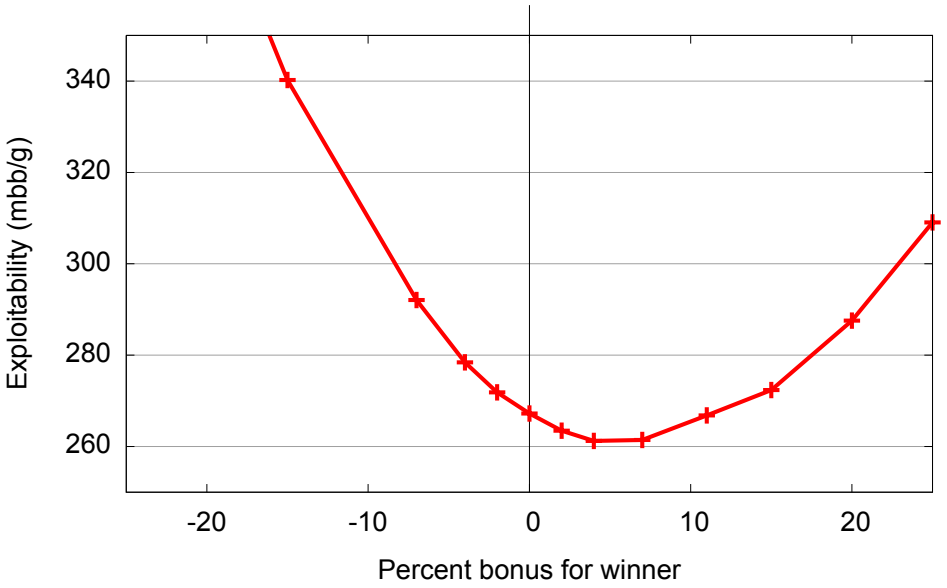


Figure 2.5: Exploitability of strategies when a bonus is added to the winner's utility.

in their space is listed for comparison.

Surprisingly, the resulting strategies were each slightly less exploitable in the untilted real game than "Pink", the abstract equilibrium. To further investigate this effect, we used the "Orange" tilt, which affects only the winner's payoffs, and varied the modification from -25% to 25% in one of our smaller new abstractions. The results of this experiment are shown in Figure 2.5. An equilibrium in this abstraction (at 0%) is exploitable for 267.235 mbb/g, while a tilt of 4% reaches 261.219 mbb/g and 7% reaches 261.425 mbb/g. One possible explanation is that the change in the resulting strategies masks some of the errors caused by the abstraction process. This is a surprising result that warrants further study.

Overfitting. Recall that the most popular approach in this domain is to minimize exploitability in an abstract game as a proxy for minimizing exploitability in the full game. Consider the counterfactual regret minimization algorithm for solving these abstract games, a popular choice among competitors. The technique iteratively improves its approximate solution, eventually converging to an unexploitable strategy in the abstract game. While we know that the exploitability is falling in the abstract game as the iterations increase, Figure 2.6 shows the exploitability in the full game for two equal-sized abstractions, as the number of iterations increases. We see that the sequence of generated strategies rapidly reduce exploitability initially, but then show a slow and steady increase in worst-case performance, all the while abstract game exploitability is decreasing. This is essentially a form of overfitting, in which the strategy's performance continues to improve in the training domain while becoming worse in its testing domain. The implications of this phenomenon deserves considerable further study.

In summary, the results that we have presented show that the poker community

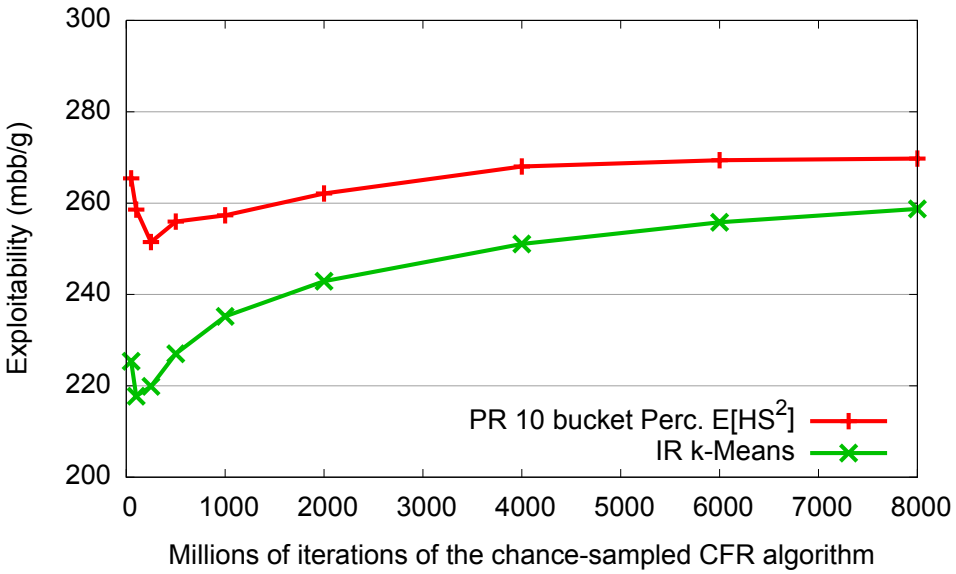


Figure 2.6: Exploitability of strategies after selected iterations of the solution technique.

has made consistent progress towards the goal of producing an unexploitable poker agent. While the least exploitable agent found so far is exploitable for 135 mbb/g, more than 2.5 times a professional's goal of 50 mbb/g, it is unlikely that an adversary that does not know the complete strategy *a priori* would be able to achieve this value. Before this work, researchers had few options to evaluate new abstraction techniques and domain features. Now, continued progress towards the goal can be measured, providing feedback to the abstraction process.

2.7 Conclusion

In this paper, we have presented a new technique that accelerates the best response calculation used to evaluate strategies in extensive games. Through this technique, we have evaluated state-of-the-art agents in the poker domain and benchmarked the community's progress towards the goal of producing an unexploitable poker agent. Our results show that there has been consistent progress towards this goal as the community discovers more efficient game solving algorithms, new abstraction techniques, and gains access to more powerful hardware. Although recent results have shown that no useful guarantees exist with the community's approach, we have now demonstrated that progress has been made.

Acknowledgments

We would like to thank Compute Canada and Westgrid for the computing resources that made this work possible. We would also like to thank Marv Andersen, Rod

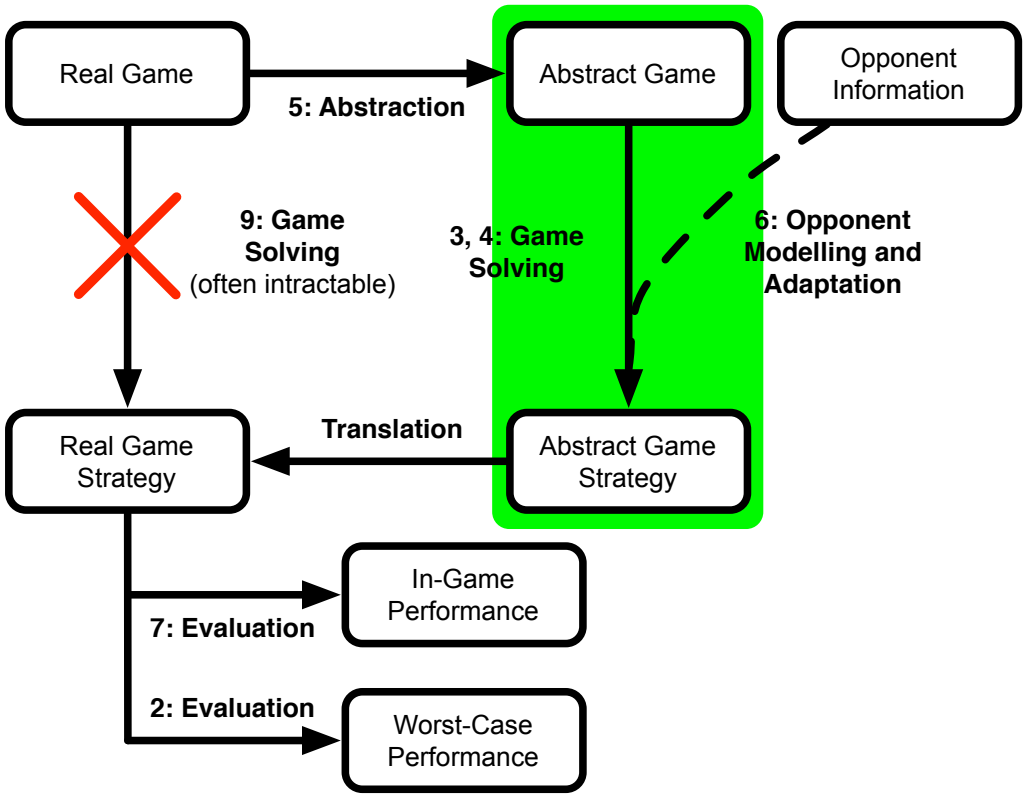
Byrnes, Mihai Ciucu, Sam Ganzfried and David Lin for their cooperation in computing the exploitability results for their poker agents. Finally, we would like to thank the Computer Poker Research Group at the University of Alberta for their helpful discussions that contributed to this work. This research was supported by NSERC and Alberta Innovates.

Chapter 3

Game Solving

Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization

Michael Johanson, Nolan Bard, Marc Lanctot,
Richard Gibson, and Michael Bowling
Presented at AAMAS 2012 [50]



The Abstraction-Solving-Translation procedure allows us to compute strong strategies for games that are too large to solve directly. However, the time and memory efficiency of our game solving algorithm remains critically important. The ability to tractably solve larger games allows us to use a finer-grained abstraction that is a better model of the real game. While there are no theoretical guarantees on improved performance, in practice larger and finer-grained strategies tend to be less exploitable and perform better in one-on-one play than their smaller, coarser cousins.

Guided by this intuition, the Annual Computer Poker Competition has helped to drive research into developing ever-more-efficient game solving algorithms. Starting in 2006, the sequence-form linear programming technique [80; 58] was used to solve coarse abstractions of subgames of heads-up limit Texas holdem, which were then merged into a complete abstract strategy. In 2007, the development of the more memory efficient Range of Skill [108] and Excessive Gap Technique [34; 41] algorithms allowed for larger four-round abstract games to be solved, and the resulting strategies had improved in-game performance.

Later in 2007, Zinkevich, Bowling, Piccione and myself introduced the Counterfactual Regret Minimization (CFR) algorithm [110], which has become the state-of-the-art technique for solving large imperfect information games. CFR is a set of self-play algorithms in which strategies are repeatedly played against each other, and then updated at each information set to minimize regret: the difference in utility between the actions chosen in earlier games, and the best action in hindsight. In the limit, the average strategy used by the players converges to a Nash equilibrium. CFR requires memory equal to twice that needed to store a strategy profile (for the "current" and "average" strategy profiles), and converges quickly and predictably towards a Nash equilibrium.

A subset of the CFR family, called Monte Carlo CFR (MCCFR) [62], allows for the use of sampling techniques while training a strategy. Instead of performing complete and precise updates during each iteration, MCCFR algorithms perform very fast and noisy updates and require many more iterations, but converge faster overall. This use of sampling is also critically important for solving abstract games, as it allows us to avoid traversing the complete real game tree on each iteration, where even one iteration may be intractable.

In this chapter's paper, we present the Public Chance Sampling CFR (PCS) algorithm. PCS was developed immediately after the Accelerated Best Response algorithm, and combines that algorithm's public tree traversal and fast terminal node evaluation with MCCFR's sampling techniques. The result is an algorithm that converges faster than the earlier CFR and MCCFR variants in games that involve a mix of public and private information, such as the public board cards and private hole cards found in poker games. In addition, PCS also became a transitional step toward descendant algorithms that offered efficiency and qualitative improvements, such as CFR-BR [49] and CFR+ [94; 17; 95], which will be discussed in Chapters 4 and 9 respectively. Another descendant, CFR-D [21], is the first imperfect information game solving algorithm that uses decomposition while retaining convergence guarantees.

Author's contributions. I developed the original idea for this paper, which is an application of the vector-based traversal and terminal node evaluation of Accelerated Best Response to the CFR algorithm. I wrote our implementation of PCS, with contributions from Bard and Gibson. I performed the experiments and interpreted the results in the poker domain. Lanctot performed the experiments in the Bluff domain, using an independent implementation of PCS which he created. The theoretical foundation of PCS is Lanctot's MCCFR algorithm, and Lanctot provided the proof of Theorem 1 presented in the appendix. All of the authors had an equal contribution in writing and editing the paper.

Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization[1]

Michael Johanson (johanson@ualberta.ca)
Nolan Bard (nolan@cs.ualberta.ca)
Marc Lanctot (lanctot@cs.ualberta.ca)
Richard Gibson (rggibson@cs.ualberta.ca)
Michael Bowling (bowling@cs.ualberta.ca)

Abstract:
Recently, there has been considerable progress towards algorithms for approximating Nash equilibrium strategies in extensive games. One such algorithm, Counterfactual Regret Minimization (CFR), has proven to be effective in two-player zero-sum poker domains. While the basic algorithm is iterative and performs a full game traversal on each iteration, sampling based approaches are possible. For instance, chance-sampled CFR considers just a single chance outcome per traversal, resulting in faster but less precise iterations. While more iterations are required, chance-sampled CFR requires less time overall to converge. In this work, we present new sampling techniques that consider sets of chance outcomes during each traversal to produce slower, more accurate iterations. By sampling only the public chance outcomes seen by all players, we take advantage of the imperfect information structure of the game to (i) avoid recomputation of strategy probabilities, and (ii) achieve an algorithmic speed improvement, performing $O(n^2)$ work at terminal nodes in $O(n)$ time. We demonstrate that this new CFR update converges more quickly than chance-sampled CFR in the large domains of poker and Bluff.

3.1 Introduction

Extensive games are an intuitive formalism for modelling interactions between agents in a sequential decision making setting. One solution concept in such domains is a Nash equilibrium. In two-player zero-sum domains, this is equivalent to a minimax strategy, which minimizes each agent's expected worst-case performance. For games of moderate size, such a strategy can be found using linear programming [58]. For larger games, techniques such as Counterfactual Regret Minimization (CFR) [110] and the Excessive Gap Technique [41] require less memory than linear programming and are capable of finding an equilibrium in games (also known as **solving** a game) with up to 10^{12} game states.

CFR is an iterative procedure that resembles self-play. On each iteration, CFR performs a full game tree traversal and updates its entire strategy profile to minimize regret at each decision. Theoretical bounds suggest that the procedure takes

[1]The paper presented in this chapter originally appeared at the *Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12)*. M. Johanson, N. Bard, M. Lanctot, R. Gibson, and M. Bowling. Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization. *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12)*, 837-844, 2012.

a number of iterations at most quadratic in the size of a player's strategy [110, Theorem 4]. Thus, as we consider larger games, not only are more iterations required to converge, but each traversal becomes more time consuming. A variant known as **Chance-Sampled (CS)** CFR [62; 110] samples one set of chance outcomes per iteration and traverses only the corresponding portion of the game tree. Compared to the basic algorithm, this sampling procedure results in faster but less precise strategy updates. In large games, the drastic reduction in per-iteration time cost outweighs the increased number of iterations required for convergence to an optimal strategy.

While CS considers only a single set of chance outcomes per iteration, recent work [53] towards fast best-response computation has shown that tree traversal and evaluation can be accelerated by simultaneously considering sets of information sets for each player. This allows for the caching and reuse of computed values, and also allows a fast terminal node evaluation in which $O(n^2)$ work can often be done in $O(n)$ time. While best response calculation in large games was previously considered intractable, the new technique was shown to perform the computation in just over one day [53].

In this paper, we apply this new tree traversal to CFR, resulting in three new sampling variants: **Self-Public Chance Sampling (SPCS)**, **Opponent-Public Chance Sampling (OPCS)**, and **Public Chance Sampling (PCS)**. The new techniques reverse the previous trend in that they advocate less sampling: a small number of slow iterations, each updating a large number of information sets, yielding precise strategy updates while reusing computed values. In particular, PCS takes advantage of the computation reuse and fast terminal node evaluation used in accelerating the best response computation. We will prove the convergence of the new techniques, investigate their qualities, and demonstrate empirically that PCS converges more quickly to an equilibrium than CS in both poker and the game of Bluff.

3.2 Background

An extensive game is a general model of sequential decision-making with imperfect information. Extensive games consist primarily of a game tree whose nodes correspond to **histories** (sequences) of actions $h \in H$. Each non-terminal history, h, has an associated **player** $P(h) \in N \cup \{c\}$ (where N is the set of players and c denotes **chance**) that selects an **action** $a \in A(h)$ at that history h. When $P(h) = c$, $f_c(a|h)$ is the (fixed) probability of chance generating action a at h. We call h a **prefix** of history h', written $h \sqsubseteq h'$, if h' begins with the sequence h. Each **terminal history** $z \in Z \subset H$ has associated **utilities** for each player i, $u_i(z)$. In **imperfect information** games, histories are partitioned into **information sets** $I \in \mathcal{I}_i$ representing different game states that player i cannot distinguish between. For example, in poker, player i does not see the opponents' private cards, and thus all histories differing only in the private cards dealt to the opponents are in the same information set for player i. For histories $h, h' \in I$, the actions available at h and h' must be the same, and we denote this action set by $A(I)$. We also assume **perfect recall** that

guarantees players always remember information that was revealed to them and the order in which it was revealed.

A **strategy for player i**, σ_i, is a function that maps each $I \in \mathcal{I}_i$ to a probability distribution over $A(I)$. We denote Σ_i as the set of all strategies for player i. A **strategy profile** is a vector of strategies $\sigma = (\sigma_1, \ldots, \sigma_{|N|})$, one for each player. We let σ_{-i} refer to the strategies in σ excluding σ_i.

Let $\pi^\sigma(h)$ be the probability of history h occurring if all players choose actions according to σ. We can decompose

$$\pi^\sigma(h) = \prod_{i \in N} \pi_i^\sigma(h) \prod_{\substack{h'a \sqsubseteq h \\ P(h')=c}} f_c(a|h')$$

into each player's and chance's contribution to this probability. Here, $\pi_i^\sigma(h)$ is the contribution from player i when playing according to σ_i. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except that of player i. Furthermore, let $\pi^\sigma(h, h')$ be the probability of history h' occurring, given h has occurred with $\pi_i^\sigma(h, h')$, and $\pi_{-i}^\sigma(h, h')$ defined similarly.

Given a strategy profile, σ, we define a player's **best response** as a strategy that maximizes their expected payoff, assuming all other players play according to σ. The **best-response value** for player i is the value of that strategy, $b_i(\sigma_{-i}) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$. A strategy profile σ is an **ϵ-Nash equilibrium** if no player can deviate from σ and gain more than ϵ; *i.e.* $u_i(\sigma) + \epsilon \geq \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$ for all $i \in N$. If $\epsilon = 0$, then σ is a **Nash equilibrium** and every player is playing a best response.

In this paper, we will focus on **two-player zero-sum** games: $N = \{1, 2\}$ and $u_1(z) = -u_2(z)$ for all $z \in Z$. In this case, the **exploitability** of σ, $\epsilon_\sigma = (b_1(\sigma_2) + b_2(\sigma_1))/2$, measures how much σ loses to a worst case opponent when players alternate positions. A Nash equilibrium has an exploitability of 0.

Lastly, define $\mathcal{C} = \{h \in H : P(h) = c\}$ to be the set of all histories where it is chance's turn to act. We will assume that $\mathcal{C}$ can be partitioned into three sets with respect to player i: $\mathcal{S}_i$, $\mathcal{O}_i$, and $\mathcal{P}$. Each set contains the histories h whose actions $a \in A(h)$, or **chance events**, are observable only by player i ($\mathcal{S}_i$), only by player i's opponent ($\mathcal{O}_i$), or by both players ($\mathcal{P}$). We refer to chance events occurring at $h \in \mathcal{S}_i \cup \mathcal{O}_i$ as **private** and to chance events occurring at $h \in \mathcal{P}$ as **public**. In addition, we assume that the actions available to the players throughout the game are independent of the private chance events. These two assumptions hold for a large class of games, including poker as well as any Bayesian game with observable actions [76] (*e.g.* , Bluff or negotiation games); furthermore, games can often be modified by adding additional chance actions to satisfy the property.

3.2.1 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) resembles a self-play algorithm where we iteratively obtain strategy profiles σ^t based on regret values accumulated throughout previous trials. At each information set $I \in \mathcal{I}_i$, the expected value for player

i at I under the current strategy is computed, assuming player i plays to reach I. This expectation is the **counterfactual value** for player i,

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

where Z_I is the set of terminal histories passing through I and $z[I]$ is the prefix of z contained in I. For each action $a \in A(I)$, these values determine the **counterfactual regrets** at iteration t, $r^t_i(I, a) = v_i(\sigma^t_{(I\to a)}, I) - v_i(\sigma^t, I)$, where $\sigma_{(I\to a)}$ is the profile σ except at I, action a is always taken. The regret $r^t_i(I, a)$ measures how much player i would rather play action a at I than play σ^t. The counterfactual regrets are accumulated and σ^t is updated by applying regret matching [40; 110] to the accumulated regrets. Regret matching is a regret minimizer; *i.e.* , over time, the average of the counterfactual regrets approaches 0. Minimizing counterfactual regret at each information set minimizes the **average overall regret** [110, Theorem 3], defined by

$$R^T_i = \max_{\sigma' \in \Sigma_i} \frac{1}{T} \sum_{t=1}^{T} \left(u_i(\sigma', \sigma^t_{-i}) - u_i(\sigma^t_i, \sigma^t_{-i}) \right).$$

It is well-known that in a two-player zero-sum game, minimizing average overall regret implies that the average profile $\overline{\sigma}^T$ is an approximate equilibrium. CFR produces an ϵ-Nash equilibrium in $O(|H||\mathcal{I}_i|/\epsilon^2)$ time [110, Theorem 4].

Rather than computing the exact counterfactual values on every iteration, one can instead sample the values using **Monte Carlo CFR (MCCFR)** [62]. **Chance-sampled (CS) CFR** [110] is an instance of MCCFR that considers just a single set of chance outcomes per iteration. In general, let $\mathcal{Q}$ be a set of subsets, or **blocks**, of the terminal histories Z such that the union of all blocks spans Z. For CS, $\mathcal{Q}$ is the partition of Z where two histories belong to the same block if and only if no two chance events differ. In addition, a probability distribution over $\mathcal{Q}$ is required and a block $Q \in \mathcal{Q}$ is sampled on each iteration, giving us the **sampled counterfactual value** for player i,

$$\tilde{v}_i(\sigma, I) = \sum_{z \in Z_I \cap Q} u_i(z)\pi^\sigma_{-i}(z[I])\pi^\sigma(z[I], z)/q(z),$$

where $q(z)$ is the probability that z was sampled. In CS, we sample the blocks according to the likelihood of the chance events occurring, so that

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C}}} f_c(a|h).$$

The counterfactual regrets are then measured according to these sampled values, as opposed to "vanilla CFR" that uses the true values $v_i(\sigma, I)$. Sampling reduces enumeration to the smaller subset Q rather than all of Z, decreasing the amount of time required per iteration. For a fixed ϵ, CS requires more iterations than vanilla CFR to obtain an ϵ-Nash equilibrium; however, the overall computing time for CS is lower in poker games [109, Appendix A.5.2].

3.2.2 Accelerated Traversal and Evaluation

A recent paper describes how to accelerate the computation of the best response value in large extensive form games [53]. This technique traverses a game's **public tree**, which represents the state of the game visible to all players. The authors observe that each player's strategy must be independent of the other player's private information. As such, a player's action probabilities can be computed just once while considering the opponent's entire set of possible private states in one traversal.

In addition, the authors describe an efficient terminal node evaluation that considers a range of n information sets for each player in tandem. If the game's payoffs exhibit structure, then it may be possible to exploit this structure and reduce a naive $O(n^2)$ computation to $O(n)$. Examples of structured payoffs include games where utilities are affected by only certain factors within the players' information sets, such as in a negotiation game, and games where information sets can be ranked from weakest to strongest, such as in poker. This algorithmic speedup is not being used in any of the previously published equilibrium solvers. In Section 3.3, we describe how to use these ideas to produce a new equilibrium solver that outperforms the current state of the art.

3.2.3 Domains: Poker and Bluff

The Game of Poker Our main poker game of interest is heads-up (*i.e.* , two-player) limit Texas hold'em poker, or simply **Texas hold'em**. The game uses a standard 52 card deck and consists of 4 betting rounds. In the first round, the **pre-flop**, each player is dealt two private cards. For subsequent rounds – in order, the **flop**, **turn**, and **river** – public community cards are revealed (3 at the flop and 1 at each of the turn and river). During each round, players sequentially take one of three actions: **fold** (forfeit the game), **call** (match the previous bet), or **raise** (increase the bet). There is a maximum of 4 raises per round, each with a fixed size, where the size is doubled on the final two rounds. If neither player folds, then the player with the highest ranked poker hand wins all of the bets.

Texas hold'em contains approximately 3.2×10^{14} information sets. The large size of the game makes an equilibrium computation intractable for all known algorithms; CFR would require more than ten petabytes of RAM and hundreds of CPU-years of computation. A common approach is to use state-space abstraction to produce a similar game of a tractable size by merging information sets or restricting the action space [39]. In Section 3.4, we consider several abstractions of Texas hold'em and two new variants of Texas hold'em that are small enough to compute equilibrium solutions using CFR without abstraction. The first new variant is **[2-1] hold'em**. The game is identical to Texas hold'em, except consists of only the first two betting rounds, the pre-flop and flop, and only one raise is allowed per round. This reduces the size of the game to 16 million information sets. Similarly, **[2-4] hold'em** has just two rounds, but the full four raises are allowed per round, resulting in 94 million information sets in total. In both [2-1] hold'em and [2-4] hold'em, the size of a raise doubles from the pre-flop to the flop.

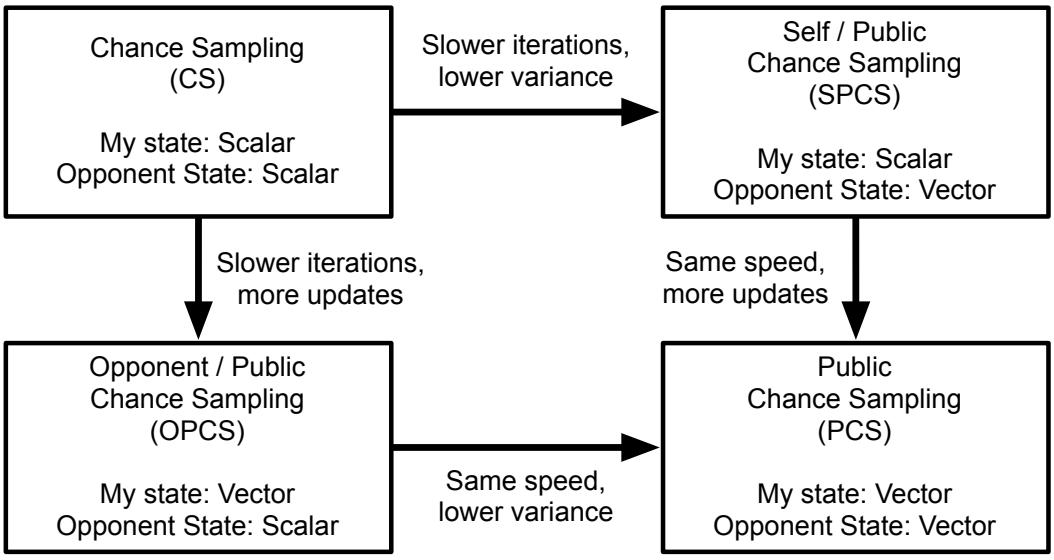


Figure 3.1: Relationship between MCCFR variants

The Game of Bluff Bluff, also known as Liar's Dice, Dudo, and Perudo, is a dice-bidding game. In our version, Bluff(D_1,D_2), each die has six sides with faces 1 to 5 and a star: $\star$. Each player i rolls D_i of these dice and looks at them without showing them to their opponent. On each round, players alternate by bidding on the outcome of all dice in play until one player claims that the other is bluffing (*i.e.* , claims that the bid does not hold). A bid consists of a **quantity** of dice and a **face** value. A face of $\star$ is considered "wild" and counts as matching any other face. For example, the bid 2-5 represents the claim that there are at least two dice with a face of 5 or $\star$ among both players' dice. To place a new bid, the player must increase either the quantity or face value of the current bid; in addition, lowering the face is allowed if the quantity is increased. The player calling bluff wins the round if the opponent's last bid is incorrect, and loses otherwise. The losing player removes one of their dice from the game and a new round begins, starting with the player who won the previous round. When a player has no more dice left, they have lost the game. A utility of $+1$ is given for a win and -1 for a loss.

In this paper, we restrict ourselves to the case where $D_1 = D_2 = 2$, a game containing 352 million information sets. Note that since Bluff(2,2) is a multi-round game, the expected values of Bluff(1,1) are precomputed for payoffs at the leaves of Bluff(2,1), which is then solved for leaf payoffs in the full Bluff(2,2) game.

3.3 New Monte Carlo CFR Variants

Before presenting our new CFR update rules, we will begin by providing a more practical description of chance-sampled CFR. On each iteration, we start by sampling all of chance's actions: the public chance events visible to each player, as well as the private chance events that are visible to only a subset of the players.

In poker, this corresponds to randomly choosing the public cards revealed to the players, and the private cards that each player is dealt. In the game of Bluff, there are no public chance events, and only private chance events are sampled for each player. Next, we recursively traverse the portion of the game tree that is reachable given the sampled chance events, and explore all of the players' actions. On the way from the root to the leaves, we pass forward two scalar values: the probability that each player would take actions to reach their respective information sets, given their current strategy and their private information. On the way back from the leaves to the root, we return a single scalar value: the sampled counterfactual value $\tilde{v}_i(\sigma, I)$ for player i. At each choice node for player i, these values are all that is needed to calculate the regret for each action and update the strategy. Note that at a terminal node $z \in Z$, it takes $O(1)$ work to determine the utility for player i, $u_i(z)$.

We will now describe three different methods of sampling chance events that have slower iterations, but do more work on each iteration. Figure 3.1 shows the relationship between CS and these three new variants, all of which belong to the MCCFR family [62] of update rules.

Opponent-Public Chance Sampling Consider a variation on CS, where instead of sampling at every chance node, we sample an action for just the opponent's chance and the public chance events while enumerating all of the possible outcomes at our private chance events. We will call this variant Opponent-Public Chance Sampling (OPCS). This can be formalized within the MCCFR framework by letting $\mathcal{Q}$ be the partition of Z such that two histories fall into the same block if and only if the actions taken at opponent and public chance events match. The probability that z is sampled is then

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{O}_i \cup \mathcal{P}}} f_c(a|h).$$

Naively, we could use the same recursive tree walk that we used for CS to perform this update, by doing one tree walk for each of our private chance outcomes in turn. However, this update allows us to traverse the sampled portion of the game tree in a much more efficient way. Since our opponent does not observe our private chance events, their strategy and choice of actions, given their single sampled chance event, cannot depend on which information set we are in. This means that we can update all of our information sets that are consistent with the current game state and the sampled public chance events at the same time, thus amortizing the cost of walking the tree over many updates. This can be achieved by a new recursive tree walk that passes forwards a vector for us (our probability of reaching the current game state with each of our private chance outcomes) and a scalar for the opponent (their probability of reaching the current game state with their single sampled private chance outcome), and returns a vector of values (our counterfactual value for each of our private chance outcomes).

At terminal nodes, we must evaluate n possible game states, each consisting of a different private chance outcome for us and one chance outcome for the opponent.

This requires $O(n)$ time. In comparison to CS, each iteration of OPCS is slower, but performs more work by updating a much larger number of information sets.

Self-Public Chance Sampling In OPCS, we enumerate over all of our possible private chance outcomes. Alternatively, we can instead enumerate over all of our opponent's private chance outcomes while sampling our own private chance outcomes and the public chance outcomes. We will call this variant Self-Public Chance Sampling (SPCS). This can similarly be formalized by defining $\mathcal{Q}$ to be the partition of Z that separates histories into different blocks whenever the actions taken at our private or public chance events differ, where

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{S}_i \cup \mathcal{P}}} f_c(a|h)$$

is the probability of sampling terminal history z.

As in OPCS, we can use an efficient recursive tree walk to perform this update. Since we cannot observe the opponent's private chance events, our strategy and choice of actions cannot depend on which information set they are in. Thus, when computing our counterfactual value, we will consider every possible private chance outcome for our opponent. Doing so forms a more accurate estimate of the true counterfactual value for our sampled outcome, compared to the noisy estimate CS and OPCS obtain through one sampled opponent private chance outcome. The SPCS tree walk passes forward a scalar for ourselves (the probability of reaching the current game state with our single chance outcome) and a vector for the opponent (their probabilities of reaching the current game state with each of their private chance outcomes), and returns a scalar (the counterfactual value for our sampled outcome).

At terminal nodes, we must evaluate up to n possible game states, formed by our single chance outcome and up to n possible chance outcomes for the opponent. This requires $O(n)$ time. In comparison to CS, each iteration is slower and performs the same number of updates to the strategy, but each update is based off of much more precise estimates.

Public Chance Sampling We will now introduce the core contribution of this work, called Public Chance Sampling (PCS), that combines the advantages of both of the previous two updates, while taking advantage of efficient terminal node evaluation to keep the time cost per iteration in $O(n)$. In PCS, we sample only the public chance events, and consider all possible private chance events for ourself and for the opponent. In other words, we define $\mathcal{Q}$ to be the partition of Z that separates histories into different blocks whenever the actions taken at a public chance event differ, where

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{P}}} f_c(a|h)$$

is the probability of sampling $z \in Z$.

PCS relies on the property that neither us nor our opponent can observe the other's private chance events, and so the action probabilities for each remain the same across the other's private information. Thus, we can perform a CFR update through a recursive tree walk with the following structure. On the way from the root to the leaves, we will pass forwards two vectors: one containing the probabilities of us and one containing the probabilities of the opponent reaching the current game state, for each player's n possible private chance outcomes. On the way back, we will return a vector containing the counterfactual value for each of our n information sets.

At the terminal nodes, we seemingly have an $O(n^2)$ computation, as for each of our n information sets, we must consider all n of the opponent's possible private outcomes in order to compute our utility for that information set. However, if the payoffs at terminal nodes are structured in some way, we can often reduce this to an $O(n)$ evaluation that returns exactly the same value as the $O(n^2)$ evaluation [53]. Doing so gives PCS the advantage of both SPCS (accurate strategy updates) and OPCS (many strategy updates) for the same evaluation cost of either.

3.3.1 Algorithm

The three new chance-sampling variants, along with CS, are shown in Algorithm 1. The WalkTree function traverses down the game tree by recursively concatenating actions, starting with the empty history $h = \emptyset$, and updates player i's regrets and average strategy on the way back up. Two vectors are maintained, one for player i, $\vec{\pi}_i$, and one for the opponent, $\vec{\pi}_{-i}$. These vectors keep track of the probabilities of reaching each information set consistent with the current history h, with each element corresponding to a different private chance outcome for that player. In CS, both vectors have length one (*i.e.* , are scalars). In OPCS, $\vec{\pi}_{-i}$ has length one because the opponent's private chance events are being sampled. Similarly, in SPCS, $\vec{\pi}_i$ has length one.

When the current sequence h is a terminal history (line 6), the utility is computed and returned. At line 7, $\vec{f}_{c,i}(h)$ and $\vec{f}_{c,-i}(h)$ are the vectors corresponding to the probability distribution over player i's and the opponent's private chance outcomes, respectively, and $\odot$ represents element-wise vector multiplication. Again, one or both vectors may have length one depending on the selected variant, in which case the single element is always 1. For OPCS and PCS, $\vec{u}_i$ is a vector containing a utility for each of player i's private outcomes; for SPCS and CS, $\vec{u}_i$ is a length one vector corresponding to the utility for player i's sampled private outcome. PCS uses the $O(n^2)$ to $O(n)$ algorithmic improvement to compute $\vec{u}_i$, which will be described in Section 3.3.2.

Chance events are handled by lines 8 to 14. When one of the four conditions at line 8 holds, we are at a chance event that is to be sampled; otherwise, we consider all possible chance events at h. In the latter case, we must take a dummy action (line 12) simply to continue traversing the tree. This action has no effect on the remainder of the tree walk due to our assumption that player actions are independent of private chance events.

Algorithm 1 PCS Algorithm

Require: a variant $v \in \{\text{CS}, \text{OPCS}, \text{SPCS}, \text{PCS}\}$.
Initialize regret tables: $\forall I, r_I[a] \leftarrow 0$.
Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.

function WALKTREE(history h, player i, reach probability $\vec{\pi}_i$, reach probability $\vec{\pi}_{-i}$)
  if $h \in Z$ **then**
    return $\vec{f}_{c,i}(h) \odot \vec{u}_i \left(h \mid \vec{\pi}_{-i} \odot \vec{f}_{c,-i}(h)\right)$
  else if $\begin{pmatrix} (v = \text{PCS and } h \in \mathcal{P}) \text{ or} \\ (v = \text{SPCS and } h \in \mathcal{S}_i \cup \mathcal{P}) \text{ or} \\ (v = \text{OPCS and } h \in \mathcal{O}_i \cup \mathcal{P}) \text{ or} \\ (v = \text{CS and } h \in \mathcal{C}) \end{pmatrix}$ **then**
    Sample outcome $a \in A(h)$ with probability $\sigma_c(a|h)$
    return WALKTREE(ha, i, $\vec{\pi}_i$, $\vec{\pi}_{-i}$)
  else if $h \in \mathcal{C}$ **then**
    Select dummy outcome $a \in A(h)$
    return WALKTREE(ha, i, $\vec{\pi}_i$, $\vec{\pi}_{-i}$)
  end if
  $\vec{I} \leftarrow$ lookupInfosets(h)
  $\vec{u} \leftarrow \vec{0}$
  $\vec{\sigma} \leftarrow$ regretMatching($\vec{I}$)
  for each action $a \in A(h)$ **do**
    if $P(h) = i$ **then**
      $\vec{\pi}'_i \leftarrow \vec{\sigma}[a] \odot \vec{\pi}_i$
      $\vec{u}' \leftarrow$ WALKTREE(ha, i, $\vec{\pi}'_i$, $\vec{\pi}_{-i}$)
      $\vec{m}[a] \leftarrow \vec{u}'$
      $\vec{u} \leftarrow \vec{u} + \vec{\sigma}[a] \odot \vec{u}'$
    else
      $\vec{\pi}'_{-i} \leftarrow \vec{\sigma}[a] \odot \vec{\pi}_{-i}$
      $\vec{u}' \leftarrow$ WALKTREE(ha, i, $\vec{\pi}_i$, $\vec{\pi}'_{-i}$)
      $\vec{u} \leftarrow \vec{u} + \vec{u}'$
    end if
  end for
  if $P(h) = i$ **then**
    for $I \in \vec{I}$ **do**
      for $a \in A(I)$ **do**
        $r_I[a] \leftarrow r_I[a] + m[a][I] - u[I]$
        $s_I[a] \leftarrow s_I[a] + \pi_i[I]\sigma[a][I]$
      end for
    end for
  end if
  return $\vec{u}$
end function

function SOLVE
  for $t \in \{1, 2, 3, \cdots\}$ **do**
    for $i \in N$ **do** WALKTREE($\emptyset$, i, $\vec{1}$, $\vec{1}$)
    end for
  end for
end function

Lines 15 to 38 handle the cases where h is a decision node for one of the players. First, lookupInfosets(h) retrieves all of the information sets consistent with h and the current player $P(h)$'s range of possible private outcomes, whether sampled ($|\vec{I}| = 1$) or not. Next, at line 17, regret matching [40; 110] determines the current strategy $\vec{\sigma}$, a vector of action probabilities for each retrieved information set (and thus, in general, a vector of vectors). Regret matching assigns action probabilities according to

$$\sigma[a][I] = \begin{cases} r_I^+[a] / \sum_{b \in A(I)} r_I^+[b] & \text{if } \sum_{b \in A(I)} r_I^+[b] > 0 \\ 1/|A(I)| & \text{otherwise,} \end{cases}$$

where $r_I^+[a] = \max\{r_I[a], 0\}$. We then iterate over each action $a \in A(h)$, recursively obtaining the expected utilities for a at each information set (line 21 or 26). When $P(h) = i$, these utilities are stored (line 22) and used to update the regret at each information set (line 33), while the current strategy $\vec{\sigma}$ weights both the returned expected utility at h (line 23) and the average strategy update (line 34). Note that at line 27, we do not weight $\vec{u}'$ by $\vec{\sigma}[a]$ since the opponent's reaching probabilities are already factored into the utility computation (line 7).

After iterating over the outer loop of Solve() (line 42) for many iterations, an ϵ-Nash equilibrium is obtained from the accumulated strategies: $\bar{\sigma}(I, a) = s_I[a] / \sum_{b \in A(I)} s_I[b]$.

3.3.2 Efficient Terminal Node Evaluation

We now describe how PCS computes a vector of expected utilities $\vec{u}_i(h \mid \vec{\pi}_{-i})$ at line 7 for player i's n private outcomes in $O(n)$ time. As we have already noted, Johanson *et al.* [53] gave a detailed description for how to do this in poker. In this section, we will describe an efficient terminal node evaluation for Bluff(D_1, D_2).

Every game ends with one player calling bluff, and the payoffs ($+1$ or -1) are determined solely by whether or not the last bid holds. Let x-y be the last such bid. We now must discriminate between cases where there are less than and where there are at least x dice showing face y or $\star$.

At the terminal history h, we have a vector of reach probabilities $\vec{\pi}_{-i}$ for each of the opponent's n possible dice rolls. Let $\vec{X}_{-i}$ be a vector of length $D_{-i} + 1$, where the element $X_{-i}[j]$ ($0 \leq j \leq D_{-i}$) equals the probability of the opponent reaching h with exactly j dice showing face y or $\star$. $\vec{X}_{-i}$ is constructed in $O(n)$ time by iterating over each element of $\vec{\pi}_{-i}$, adding the probability to the appropriate entry of $\vec{X}_{-i}$ at each step. We can then compute the expected utility for player i with exactly j of his or her dice showing face y or $\star$. If player i called bluff, this expected utility is

$$U_i[j] = \sum_{\ell=0}^{x-j-1} (+1) \cdot X_{-i}[\ell] + \sum_{\ell=x-j}^{D_{-i}} (-1) \cdot X_{-i}[\ell];$$

if the opponent called bluff, the expected utility is $-U_i[j]$. Constructing $\vec{U}_i$ takes $O(n)$ time. Finally, we iterate over all $k \in \{1, ..., n\}$ and set $u_i[k] = U_i[x_k]$, where

x_k is the number of dice showing face y or $\star$ in player i's k^{th} private outcome. In total, the process takes $3O(n) = O(n)$ time.

3.3.3 Theoretical Analysis

CS, OPCS, SPCS, and PCS all belong to the MCCFR family of algorithms. As such, we can apply the general results for MCCFR to obtain a probabilistic bound on the average overall regret for CS and our new algorithms. Recall that in a two-player zero-sum game, minimizing average overall regret produces an ϵ-Nash equilibrium. The proof of Theorem 1 is in the appendix.

Theorem 1 *For any $p \in (0, 1]$, when using CS, OPCS, SPCS, or PCS, with probability at least $1 - p$, the average overall regret for player i is bounded by*

$$R_i^T \leq \left(1 + \frac{2}{\sqrt{p}}\right) \frac{\Delta_{u,i} M_i \sqrt{A_i}}{\sqrt{T}},$$

where M_i is a property of the game satisfying
$\sqrt{|\mathcal{I}_i|} \leq M_i \leq |\mathcal{I}_i|$, $\Delta_{u,i} = max_{z,z'}|u_i(z) - u_i(z')|$, and $A_i = \max_{I \in \mathcal{I}_i} |A(I)|$.

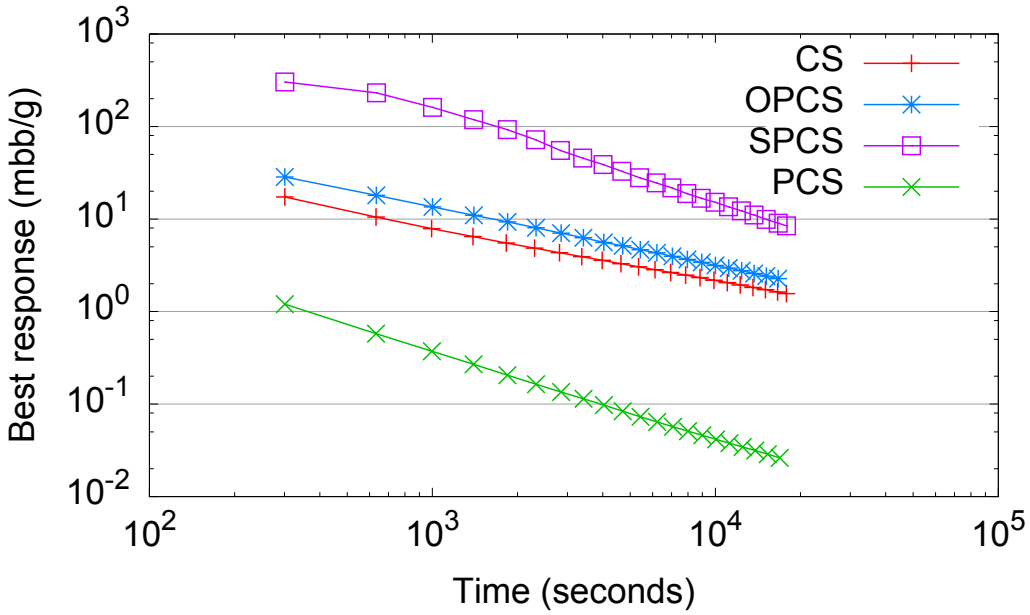
3.4 Results

The efficacy of these new updates are examined through an empirical analysis in both poker and Bluff. We begin the analysis by examining the performance of CS, SPCS, OPCS and PCS in two small games, [2-1] hold'em and [2-4] hold'em. We will then present the performance of CS and PCS in a set of Texas hold'em abstract games, to investigate their usefulness under the conditions of the Annual Computer Poker Competition. Finally, we will apply CS and PCS to the Bluff domain.

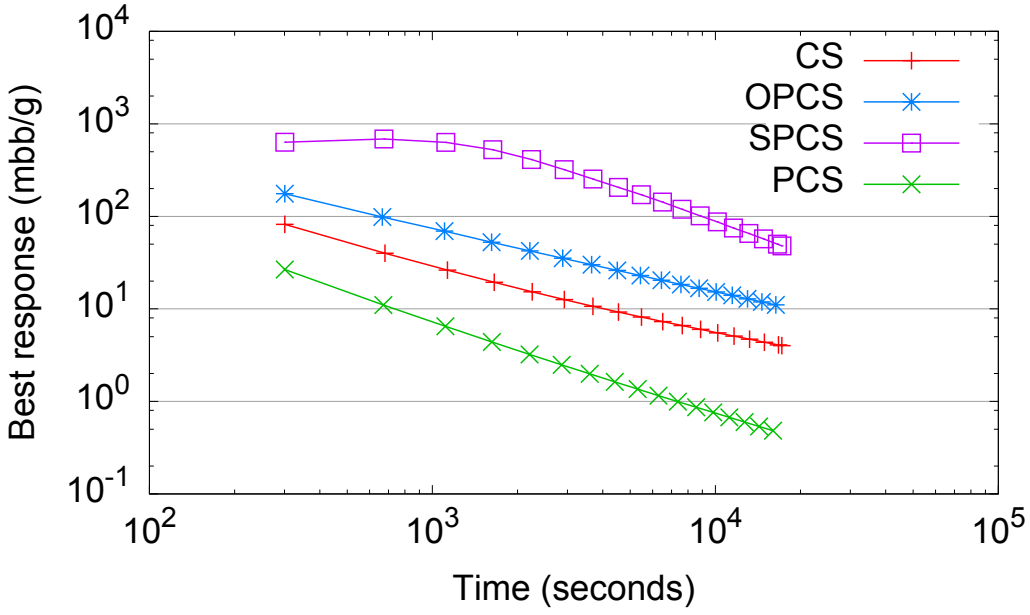
Poker [2-1] hold'em and [2-4] hold'em are games that are small enough to be tractably solved using all four of the CFR variants we are investigating: CS, SPCS, OPCS and PCS. As discussed in Section 3.3, SPCS, OPCS and PCS all perform $O(n)$ work at each terminal state, and are thus of comparable speed. However, all three require more time per iteration than CS, and to converge faster than CS, the advantage of each approach (more precise updates, more work per iteration, or both) must overcome this speed penalty.

Figure 3.2 shows the convergence of CS, OPCS, SPCS and PCS towards an optimal strategy in these small hold'em variants. We see that SPCS and OPCS converge slower than CS; the difference in speed is too great for the higher quality iterations. However, we find that PCS converges much more quickly than CS in these small games.

While [2-1] hold'em and [2-4] hold'em can be tractably solved using CFR, solving the much larger game of Texas hold'em is intractable. A common procedure used by competitors in the Annual Computer Poker Competition is to use a state-space abstraction technique to produce a smaller, similar game that can be tractably



(a) $[2-1]$ hold'em, 16 million information sets



(b) $[2-4]$ hold'em, 94 million information sets

Figure 3.2: Log-log graphs displaying convergence of best response values over time for different CFR update methods in two small unabstracted hold'em like poker games. Best response values are in milli-big-blinds per game (mbb/g). Each curve shows the average performance over five independent runs.

solved, and the resulting **abstract strategy** can then be used to select actions in the original game. The abstract strategy is an ϵ-Nash equilibrium in the abstract game, and we can measure its rate of convergence by calculating a best response within the abstract game. A critical choice in this procedure is the granularity of the abstraction. In practice, larger and finer-grained abstractions take longer to solve, but result in better approximations to a Nash equilibrium [53].

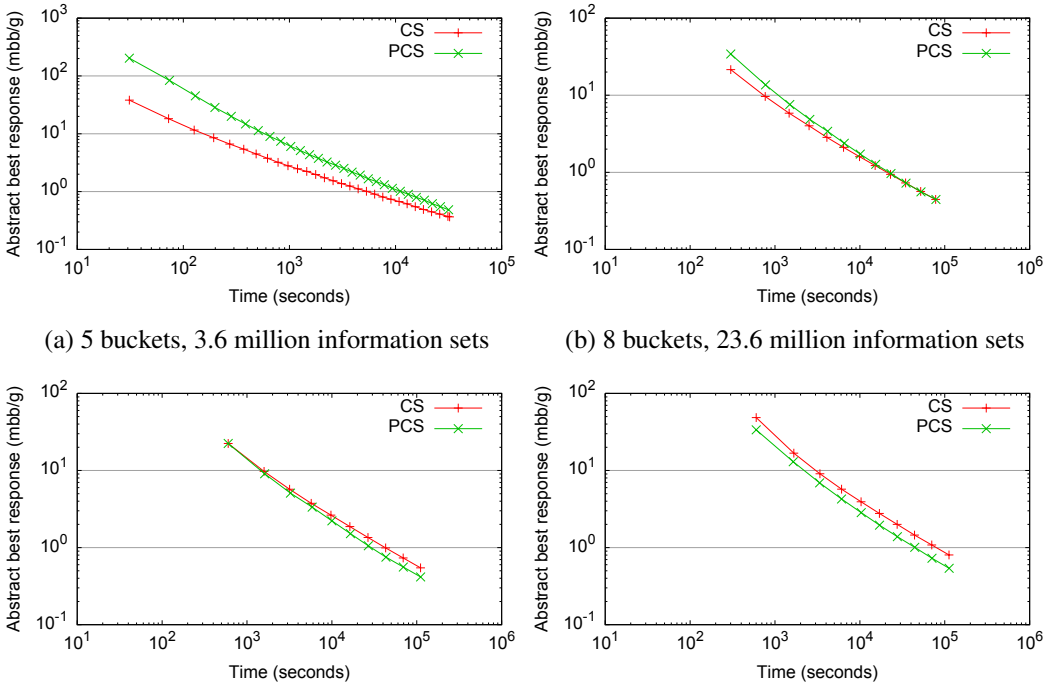
In Figure 3.3, we apply the CS and PCS algorithms to four sizes of abstract Texas hold'em games. The abstraction technique used in each is Percentile $E[HS^2]$, as described in [110], which merges information sets together if the chance events assign similar strength to a player's hand. An n-bucket abstraction branches the chance outcomes into n categories on each round.

In the smallest abstract game in Figure 3.3a, we find that CS converges more quickly than PCS. As we increase the abstraction granularity through Figures 3.3b, 3.3c and 3.3d, however, we find that PCS matches and then surpasses CS in the rate of convergence. In each of these games, the chance sampling component samples outcomes in the real game and then maps this outcome to its abstract game equivalent. When a small abstraction is used, this means that many of the information sets being updated by PCS in one iteration will share the same bucket, and some of the benefit of updating many information sets at once is lost. In larger abstract games, this effect is diminished and PCS is of more use.

In the Annual Computer Poker Competition, many competitors submit entries that are the result of running CFR on very large abstract games. Computing a best response within such abstractions, as we did in Figure 3.3, is often infeasible (as many competitors use abstractions with imperfect recall). In these circumstances, we can instead evaluate a strategy based on its performance in actual games against a fixed opponent. We can use this approach to evaluate the strategies generated by CS and PCS at each time step, to investigate how PCS and CS compare in very large games.[2]

The results of this experiment are presented in Figure 3.4. The opponent in each match is Hyperborean 2010.IRO, which took third place in the 2010 Annual Computer Poker Competition's heads-up limit Texas hold'em instant runoff event. The y-axis shows the average performance in milli-big-blinds per game (mbb/g) over a 10-million hand match of duplicate poker, and the results are accurate to ± 1 mbb/g (so the difference in curves is statistically significant). The abstraction used for CS and PCS in this experiment uses imperfect recall and has 880 million information sets, and is similar to but slightly larger than Hyperborean's abstraction, which contains 798 million information sets. At each time step, the strategies produced by PCS perform better against Hyperborean than those produced by CS. Consider the horizontal difference between points on the curves, as this indicates the additional amount of time CS requires to achieve the same performance as PCS. As the competition's winner is decided based on one-on-one performance, this result suggests

[2] Another possible evaluation metric is to compute the real-game exploitability of the strategies. However, the overfitting effect described in [53] makes the results unclear, as a strategy can become more exploitable in the real game as it approaches an equilibrium in the abstract game.



(a) 5 buckets, 3.6 million information sets

(b) 8 buckets, 23.6 million information sets

(c) 10 buckets, 57.3 million information sets

(d) 12 buckets, 118.6 million information sets

Figure 3.3: Log-log graphs displaying convergence of abstract best response values over time for different CFR update methods in two perfect recall abstractions of heads-up limit Texas hold'em poker. Best response values are in milli-big-blinds per game (mbb/g). Each curve shows the average performance over five independent runs.

that PCS is an effective choice for creating competition strategies.

Bluff Bluff(2,2) is small enough that no abstraction is required. Unlike poker, all of the dice rolls are private and there are no public chance events. In this domain, one iteration of PCS is equivalent to a full iteration of vanilla CFR (*i.e.* , no sampling). However, the reordering of the computation and the fast terminal node evaluation allows PCS to perform the iteration more efficiently than vanilla CFR. Figure 3.5 shows the convergence rates of CS and PCS in Bluff on a log-log scale. We notice that PCS converges towards equilibrium significantly faster than CS does. As noted earlier, PCS has two speed advantages: the fast terminal node evaluation, and the ability to reuse the opponent's probabilities of reaching an information set for many of our own updates. By comparison, vanilla CFR would traverse the action space 441 times to do the work of 1 PCS traversal. Similar to Figure 3.4 in the poker experiments, we can also compare the performance of CS and PCS strategies against a fixed opponent: an ϵ-Nash equilibrium for Bluff(2,2). This experiment is presented in Figure 3.6, and the fixed opponent is the final data point of the PCS line; the results are similar if the final CS data point is used. This result shows that PCS is also more efficient than CS at producing effective strategies for one-on-one matches.

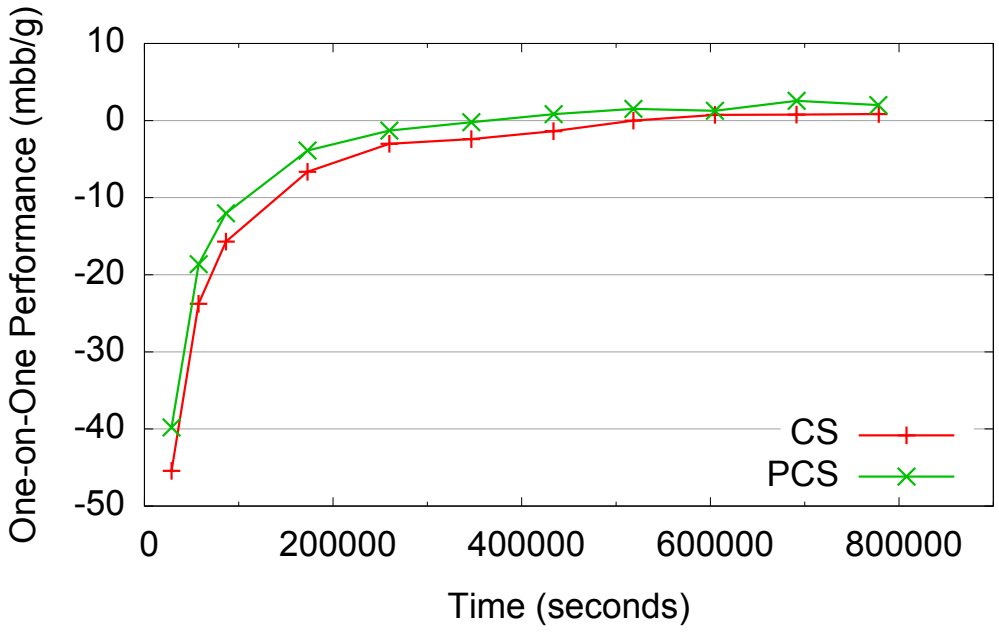


Figure 3.4: Performance of CS and PCS strategies in a large abstraction against a fixed, strong opponent.

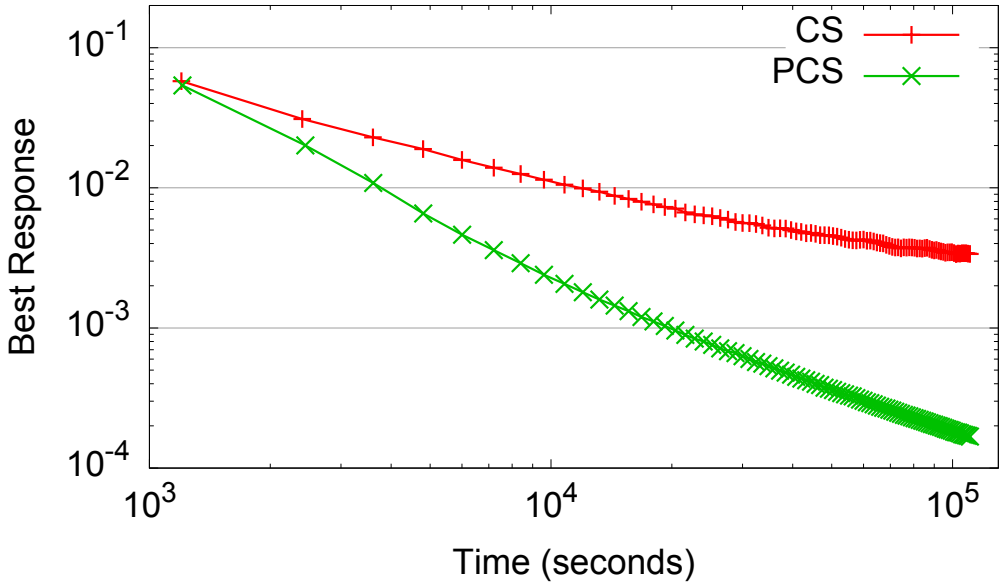


Figure 3.5: Log-log graph showing convergence of CS and PCS towards an equilibrium in Bluff(2,2). Each curve shows the average performance over five independent runs.

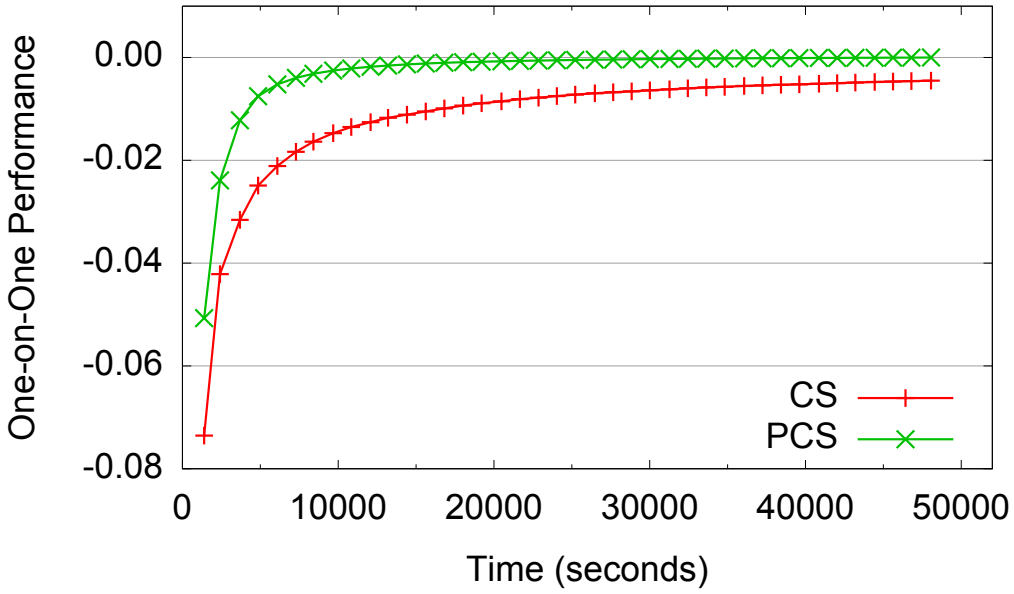


Figure 3.6: Performance of CS and PCS strategies against an ϵ-Nash equilibrium in Bluff(2,2)

3.5 Conclusion

Chance Sampled CFR is a state-of-the-art iterative algorithm for approximating Nash equilibria in extensive form games. In this work, we presented three new CFR variants that perform less sampling than the standard approach. They perform slower but more efficient and precise iterations. We empirically demonstrated that Public Chance Sampling converges faster than Chance Sampling on large games, resulting in a more efficient equilibrium approximation algorithm demonstrated across multiple domains. Future work will look to tighten the theoretical bounds on the new algorithms to prove that they can outperform Chance Sampling.

Acknowledgments

The authors would like to thank the members of the Computer Poker Research Group at the University of Alberta for helpful conversations pertaining to this research. This research was supported by NSERC, Alberta Innovates Technology Futures, and the use of computing resources provided by WestGrid and Compute Canada.

3.6 Appendix

Proof of Theorem 1 Let $\vec{a}_i$ be a subsequence of a history such that it contains only player i's actions in that history, and let $\vec{A}_i$ be the set of all such subsequences.

Let $\mathcal{I}_i(\vec{a}_i)$ be the set of all information sets where player i's action sequence up to that information set is $\vec{a}_i$. Without loss of generality, assume $i = 1$. Let $\mathcal{D} = \mathcal{C}, \mathcal{O}_1 \cup \mathcal{P}, \mathcal{S}_1 \cup \mathcal{P}$, or $\mathcal{P}$ depending on whether we are using CS, OPCS, SPCS, or PCS respectively. The probability of sampling terminal history z is then

$$q(z) = \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{D}}} f_c(a|h). \tag{3.1}$$

Let $\vec{a}_i \in \vec{A}_i$, $B = \mathcal{I}_i(\vec{a}_i)$, and let $Q \in \mathcal{Q}$. By [63, Theorem 7], it suffices to show that

$$Y = \sum_{I \in B} \left(\sum_{z \in Z_I \cap Q} \pi^\sigma_{-1}(z[I]) \pi^\sigma(z[I], z)/q(z) \right)^2 \leq 1.$$

By (3.1) and definition of π^σ_{-i}, we have

$$Y = \sum_{I \in B} \left(\sum_{z \in Z_I \cap Q} \pi^\sigma_2(z[I]) \pi^\sigma_{1,2}(z[I], z) \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C} \setminus \mathcal{D}}} f_c(a|h) \right)^2. \tag{3.2}$$

Now by the definition of $\mathcal{Q}$, for each $h \in \mathcal{D}$, there exists a unique $a^*_h \in A(h)$ such that if $z \in Q$ and $h \sqsubseteq z$, then $ha^*_h \sqsubseteq z$. Next, we define a new probability distribution on chance events according to

$$\hat{f}_c(a|h) = \begin{cases} 1 & \text{if } h \in \mathcal{D}, a = a^*_h \\ 0 & \text{if } h \in \mathcal{D}, a \neq a^*_h \\ f_c(a|h) & \text{if } h \in \mathcal{C} \setminus \mathcal{D}. \end{cases}$$

Notice that $\prod_{ha \sqsubseteq z, h \in \mathcal{D}} \hat{f}_c(a|h)$ is 1 if $z \in Q$ and is 0 if $z \notin Q$. Thus from (3.2), we have

$$\begin{aligned} Y &= \sum_{I \in B} \left(\sum_{z \in Z_I} \pi^\sigma_2(z[I]) \pi^\sigma_{1,2}(z[I], z) \prod_{\substack{ha \sqsubseteq z \\ h \in \mathcal{C}}} \hat{f}_c(a|h) \right)^2 \\ &= \sum_{I \in B} \left(\sum_{z \in Z_I} \hat{\pi}^\sigma_{-1}(z[I]) \hat{\pi}^\sigma(z[I], z) \right)^2 \\ &\quad \text{where } \hat{\pi}^\sigma \text{ is } \pi^\sigma \text{ except } f_c \text{ is replaced by } \hat{f}_c \\ &= \sum_{I \in B} \left(\sum_{h \in I} \hat{\pi}^\sigma_{-1}(h) \sum_{z \in Z_I} \hat{\pi}^\sigma(h, z) \right)^2 \\ &= \sum_{I \in B} \left(\sum_{h \in I} \hat{\pi}^\sigma_{-1}(h) \right)^2 \leq 1, \end{aligned}$$

where the last inequality follows by [63, Lemma 16]. □.

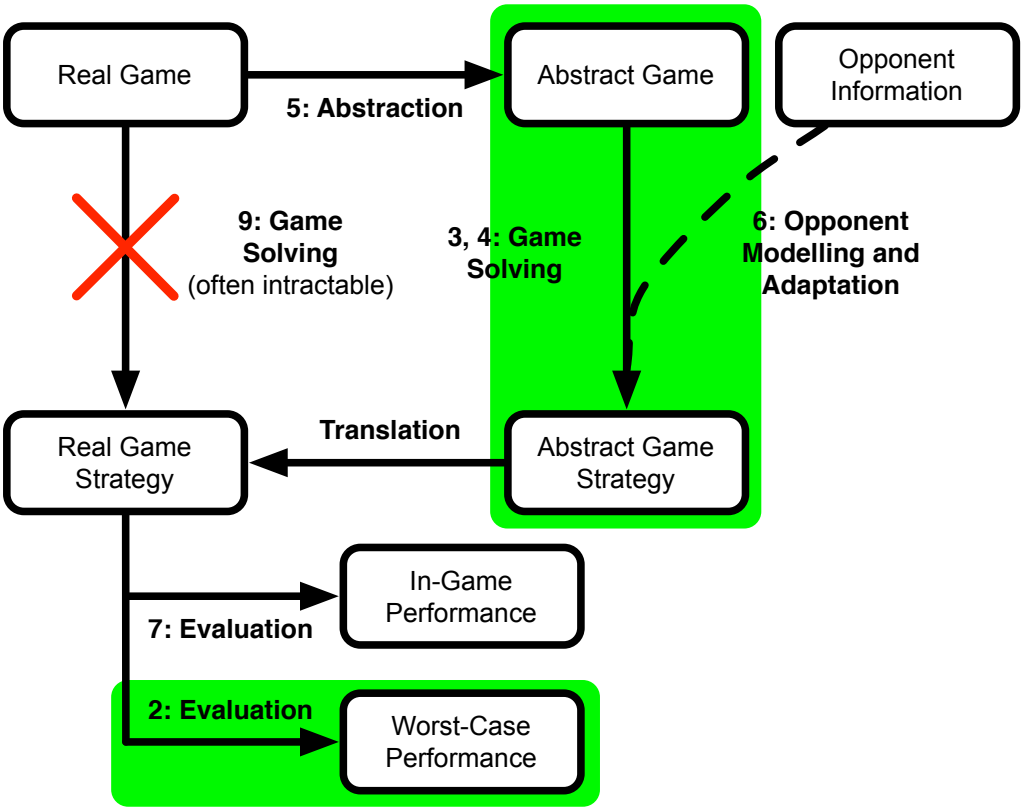
Chapter 4

Optimal Abstract Strategies

Finding Optimal Abstract Strategies
in Extensive-Form Games

Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling

Presented at AAAI 2012 [49]



The Abstraction-Solving-Translation procedure lets us create strategies for large games, but with no theoretical guarantees on their real game exploitability. In addition to the anticipated errors from using lossy abstraction techniques, the abstraction pathologies discovered by Waugh *et al.* [103] and the overfitting effect [53] described in Chapter 2 mean that an abstract game Nash equilibrium may not be close to a real game Nash equilibrium, and is unlikely to even be the least exploitable strategy that can be represented within the abstraction. This raises a question: if abstract equilibria are not the least exploitable strategies, then how can we evaluate the quality of our abstractions? How closely can our abstractions approximate a real Nash equilibrium?

In their work on abstraction pathologies, Waugh *et al.* noted that if a game could be solved where one player used abstraction and their opponent did not, then the abstracted player's strategy would have the lowest exploitability of any strategy within that abstraction, and thus be immune to abstraction pathologies and the overfitting effect. Given the immense memory and computational cost of solving a game with even one unabstracted player, this property was thought to only be of theoretical interest. However, unabstracted opponent strategies are not *always* infeasible to compute, as the Accelerated Best Response algorithm presented in Chapter 2 does exactly that. This insight helped guide us to a new game solving algorithm that uses abstraction while avoiding abstraction pathologies and overfitting entirely.

In this chapter's paper we present the CFR-BR algorithm, which combines the Public Chance Sampling (CFR) and Accelerated Best Response (BR) algorithms. As in other CFR variants, the algorithm is iterative and resembles two strategies competing against each other. In this variant, one player updates their abstracted strategy using CFR with the vector-style updates used in PCS, while their unabstracted opponent computes and uses a best response on every iteration. While performing a 76 CPU-day best response computation *inside a loop* might at first seem infeasible, the algorithm can be implemented to converge in quite reasonable time, and represents the unabstracted opponent while using less memory than PCS would need for an abstracted opponent. The result is not an abstract game Nash equilibrium, but an **optimal abstract strategy**: one with the lowest real game exploitability that can be represented in the abstract space.

Author's contributions. I was responsible for the idea behind CFR-BR, which was to combine PCS and Accelerated Best Response to solve a game with an unabstracted opponent, and thus avoid abstraction pathologies and overfitting effects. I also developed the steps presented in Figure 4.2 that make it feasible to compute CFR-BR. I created our implementation of CFR-BR, with assistance from Bard. Bowling and Burch contributed the theoretical foundation. In particular, Burch provided the proof of Theorems 4 and 5. I performed all of the experiments and interpreted the results. The paper was written and edited equally by all four authors.

Finding Optimal Abstract Strategies in Extensive-Form Games[1][2]

Michael Johanson (johanson@ualberta.ca)
Nolan Bard (nbard@ualberta.ca)
Neil Burch (burch@ualberta.ca)
Michael Bowling (mbowling@ualberta.ca)

Abstract:
Extensive-form games are a powerful model for representing interactions between agents. Nash equilibrium strategies are a common solution concept for extensive-form games and, in two-player zero-sum games, there are efficient algorithms for calculating such strategies. In large games, this computation may require too much memory and time to be tractable. A standard approach in such cases is to apply a lossy state-space abstraction technique to produce a smaller abstract game that can be tractably solved, while hoping that the resulting abstract game equilibrium is close to an equilibrium strategy in the unabstracted game. Recent work has shown that this assumption is unreliable, and an arbitrary Nash equilibrium in the abstract game is unlikely to be even near the least suboptimal strategy that can be represented in that space. In this work, we present for the first time an algorithm which efficiently finds optimal abstract strategies — strategies with minimal exploitability in the unabstracted game. We use this technique to find the least exploitable strategy ever reported for two-player limit Texas hold'em.

4.1 Introduction

Extensive-form games are a general model of multiagent interaction. They have been used to model a variety of scenarios including game playing [110; 62; 41; 79], bargaining and negotiation [64; 30], argumentation [77], and even distributed database management [70]. Strategic reasoning in all but the simplest such models has proven computationally challenging beyond certain special cases. Even the most theoretically-straightforward setting of two-player, zero-sum extensive-form games presents obstacles for finding approximate solutions for human-scale interactions (e.g., two-player, limit Texas hold'em with its 10^{18} game states). These obstacles include the recently discovered existence of abstraction pathologies [103] and a form of abstract game "overfitting" [53]. This paper presents the first technique for overcoming these abstraction challenges in the two-player, zero-sum setting.

Abstraction, first suggested by Billings and colleagues [11], is the dominant approach for handling massive extensive-form imperfect information games and

[1]The paper presented in this chapter originally appeared at the *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*. M. Johanson, N. Bard, N. Burch, and M. Bowling. Finding Optimal Abstract Strategies in Extensive-Form Games. *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 1371-1379, 2012.

[2]The appendix has additional figures that extend the results presented here.

is used by the majority of top competitors in the Annual Computer Poker Competition [84]. The approach involves constructing an abstract game by aggregating each player's states (*i.e.* , information sets) into abstract game states [39; 110]. An ϵ-Nashequilibrium is computed in the abstract game, and that strategy is then employed in the original game. As equilibrium computation algorithms improve or computational resources become available, a refined, less abstract but larger, game can be solved instead. This improvement, as larger and larger abstract games are solved, has appeared to drive much of the advancement in the Annual Computer Poker Competitions [84].

However, recent work by Waugh *et al.* [103] showed that solving more refined abstractions is not always better by presenting examples of **abstraction pathologies** in toy poker games. They showed that even when considering strict refinements of an abstraction (*i.e.* , one capable of representing a strictly larger set of strategies), the equilibria found in this finer-grained abstraction could be dramatically worse approximations than equilibria in the coarser abstraction. Furthermore, their experiments showed that while an abstraction may be able to represent good approximations of real game equilibria, these good abstract strategies may not be abstract game equilibria.

A recent publication presented a technique for efficiently computing best-responses in very large extensive-form games [53]. This made it possible to investigate Waugh's findings in the context of full two-player limit Texas hold'em. While abstraction pathologies were not found to be common using typical abstraction techniques, it was discovered that equilibrium learning methods, such as Counterfactual Regret Minimization (CFR) [110], can "overfit": as the approximation gets more exact in the abstract game, its approximation of the full-game equilibrium can worsen (see Figure 4.1).

Combined, these results present a rather bleak picture. It is unclear how to use more computational power to better approximate a Nash equilibrium in massive extensive-form games. Furthermore, our current abstractions are likely able to represent better approximations than our current methods actually compute. In this paper, we present the first algorithm that avoids abstraction pathologies and overfitting entirely. Essentially, the approach leaves one player unabstracted and finds the best possible abstract strategy for the other player. It avoids the memory requirements for solving for an unabstracted opponent by having the opponent employ a best-response strategy on each iteration rather than a no-regret strategy. It then uses sampling tricks to avoid the computational requirements needed to compute an exact best-response on each iteration. The resulting algorithm, CFR-BR, finds optimal abstract strategies, i.e, the best-approximation to a Nash equilibrium that can be represented within a chosen strategy abstraction. Consequently, it is not subject to abstraction pathologies or overfitting. We demonstrate the approach in two-player limit Texas hold'em, showing that it indeed finds dramatically better Nash equilibrium approximations than CFR with the same abstraction. We use the technique to compute the least exploitable strategy ever reported for this game.

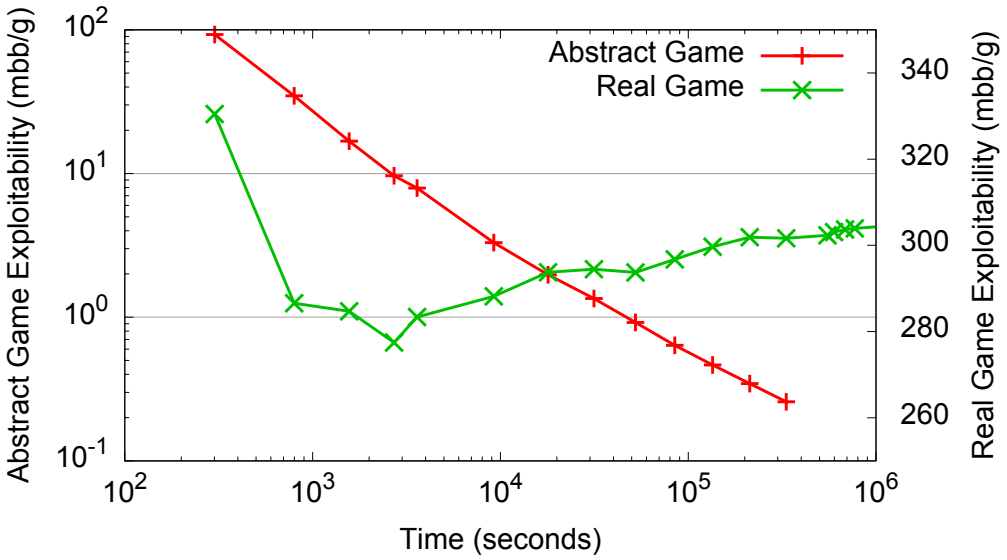


Figure 4.1: Abstract-game and real-game exploitability of strategies generated by the CFR algorithm.

4.2 Background

We begin with some formalism for extensive-form games and the counterfactual regret minimization algorithm.

Extensive-Form Games For a complete description see [76]. Extensive-form games provide a general model for domains with multiple agents making decisions sequentially. They can be viewed as a game tree that consists of nodes corresponding to histories of the game and edges between nodes being actions taken by agents or by the environment. Therefore each **history** $h \in H$ corresponds to a past sequence of actions from the set of **players**, N, and **chance**, c. For each non-terminal history h, the acting player $P(h) \in N \cup \{c\}$ selects an **action** a from $A(h)$, the set of actions available at h. We call h' a **prefix** of h, written as $h' \sqsubseteq h$, if h begins with h'. Each terminal history $z \in Z$ has a **utility** associated with it for each player i, $u_i(z)$. If $\sum_{i \in N} u_i(z) = 0$ then the game is **zero-sum**. This work focuses on two-player, zero-sum games (*i.e.* , $u_1(z) = -u_2(z)$). Let $\Delta_i = \max_{z \in Z} u_i(z) - \min_{z \in Z} u_i(z)$, be the range of utilities for player i. In our case, a two-player zero-sum game, Δ_i is the same for both players and so we refer to it simply as Δ.

In **imperfect information** games, actions taken by the players or by chance may not be observable by all of the other players. Extensive games model imperfect information by partitioning the histories where each player acts into **information sets**. For each information set $I \in \mathcal{I}_i$, player i cannot distinguish between the histories in I. It is required that $A(h)$ must equal $A(h')$ for all $h, h' \in I$, so we can denote the actions available at an information set as $A(I)$. Furthermore, we

generally require the information partition to satisfy **perfect recall**, i.e., all players are able to distinguish histories previously distinguishable or in which they took a different sequence of actions. Poker is an example of an imperfect information game since chance acts by dealing cards privately to the players. Since player i cannot see the cards of the other players, histories where only the cards of i's opponents differ are in the same information set.

A **strategy** for player i, $\sigma_i \in \Sigma_i$, maps each information set $I \in \mathcal{I}_i$ to a probability distribution over the actions $A(I)$. The **average strategy**, $\bar{\sigma}_i^t$, of the strategies $\sigma_i^1, \ldots, \sigma_i^t$ defines $\bar{\sigma}_i^t(I)$ as the average of $\sigma_i^1(I), \ldots, \sigma_i^t(I)$ weighted by each strategy's probability of reaching I [110, Equation 4]. A **strategy profile**, $\sigma \in \Sigma$, is a vector of strategies $(\sigma_1, \ldots, \sigma_{|N|})$. We let σ_{-i} refer to the strategies in σ except for σ_i. Given a strategy profile, we define player i's **expected utility** as $u_i(\sigma)$ or, since we are using two-player games, $u_i(\sigma_1, \sigma_2)$. We define $b_i(\sigma_{-i}) = \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i})$ to be the **best response value** for player i against their opponents σ_{-i} (a **best response** is the argmax). A strategy profile σ is an **ϵ-Nash equilibrium** if no player can gain more than ϵ by unilaterally deviating from σ. That is, if $b_i(\sigma_{-i}) \leq u_i(\sigma_i, \sigma_{-i}) + \epsilon$, for all $i \in N$. If this holds when $\epsilon = 0$, then all players are playing a best response to σ_{-i}, and this is called a **Nash equilibrium**. In two-player zero-sum games, we define the **game value**, v_i, for each player i to be the unique value of $u_i(\sigma^*)$ for any Nash equilibrium profile σ^*. Finally, in two-player zero-sum games we define $\varepsilon_i(\sigma_i) = b_{-i}(\sigma_i) - v_{-i}$ to be the **exploitability** of strategy σ_i, and $\varepsilon(\sigma) = (\varepsilon_1(\sigma_1) + \varepsilon_2(\sigma_2))/2 = (b_1(\sigma_2) + b_2(\sigma_1))/2$ to be the exploitability (or best response value) of the strategy profile σ. This measures the quality of an approximation to a Nash equilibrium profile, as Nash equilibria have an exploitability of 0.

Counterfactual Regret Minimization CFR [110] is a state-of-the-art algorithm for approximating Nash equilibria in two-player, zero-sum, perfect-recall games. It is an iterative algorithm that resembles self-play. Two strategies, one for each player, are represented in memory and initialized arbitrarily. In each iteration, the strategies are evaluated with respect to each other and updated so as to minimize a weighted form of regret at each decision: the difference in utility between the actions currently being selected and the best action in retrospect. Over a series of iterations, the average strategy for the players approaches a Nash equilibrium. As our algorithm builds upon CFR, we will restate some theory and formalism from that work.

Define R_i^T, player i's **average overall regret** over T steps, as $R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t))$. In other words, average overall regret is how much more utility a player could have attained on average had they played some other static strategy instead of the sequence of strategies they actually played.

Theorem 2 *(Folk theorem used by Zinkevich et al. [110, Theorem 2]) In a two-player zero-sum game at time T, if $R_i^T < \epsilon_i$ for both players, then $\bar{\sigma}^T$ is an $(\epsilon_1 + \epsilon_2)$-Nash equilibrium.*

Theorem 3 *[110, Theorem 4] If player i is updating their strategy with CFR, then* $R_i^T \leq \Delta|\mathcal{I}_i|\sqrt{|A_i|}/\sqrt{T}$ *where* $|A_i| = \max_{I \in \mathcal{I}} |A(I)|$

Since Theorem 3 bounds R_i^T, it follows from Theorem 2 that both players playing according to CFR will yield an average strategy $\bar{\sigma}^T$ that is an $(\epsilon_1 + \epsilon_2)$-Nash equilibrium where $\epsilon_i = \Delta|\mathcal{I}_i|\sqrt{|A_i|}/\sqrt{T}$.

4.3 CFR-BR

In Waugh and colleagues' work on abstraction pathologies, they found one case in which abstraction pathologies do not occur [103, Theorem 3]. When solving a game where one agent uses abstraction and the other does not, Waugh *et al.* noted that a strict refinement to the abstraction will result in a monotonic decrease in the abstracted player's exploitability. In addition, we note that the abstracted player's strategy in this equilibrium is by definition the least exploitable strategy that can be represented in the space; otherwise, it would not be an equilibrium. Thus, applying an iterative algorithm such as CFR to this asymmetrically abstracted game will avoid both the pathologies and the overfitting problem, as convergence towards the equilibrium directly minimizes exploitability. However, Waugh *et al.* [103, Page 4] note that "...solving a game where even one player operates in the null abstraction is typically infeasible. This is certainly true in the large poker games that have been examined recently in the literature."

We will now present an algorithm that achieves exactly this goal – solving a game where the opponent is unabstracted – and we will demonstrate the technique in the large domain of two-player limit Texas hold'em poker, just such a poker game which has been examined recently in the literature. Our technique, called **CFR-BR**, does this without having to explicitly store the unabstracted opponent's entire strategy, and thus avoids the large memory requirement for doing so. Our explanation of CFR-BR involves two steps, and is illustrated in Figure 4.2. For our explication, we will assume without loss of generality that the abstracted player is player 1, while the unabstracted player is player 2.

Training against a Best Response We begin by presenting an alternative method for creating the unabstracted opponent's strategy. The proof of CFR's convergence relies on the folk theorem presented as Theorem 2. Using CFR to update a player's strategy is just one way to create a regret minimizing agent needed to apply the theorem. A best response is also a regret minimizing agent, as it will achieve at most zero regret on every iteration by always choosing the highest valued actions. We will call an agent with this strategy update rule a **BR-agent**, and its strategy on any iteration will be a best response to its opponent's strategy on that same iteration.[3]

[3]Note that we could not employ two BR-agents in self-play, as they would each have to be a best-response to each other, and so a single iteration would itself require solving the game.

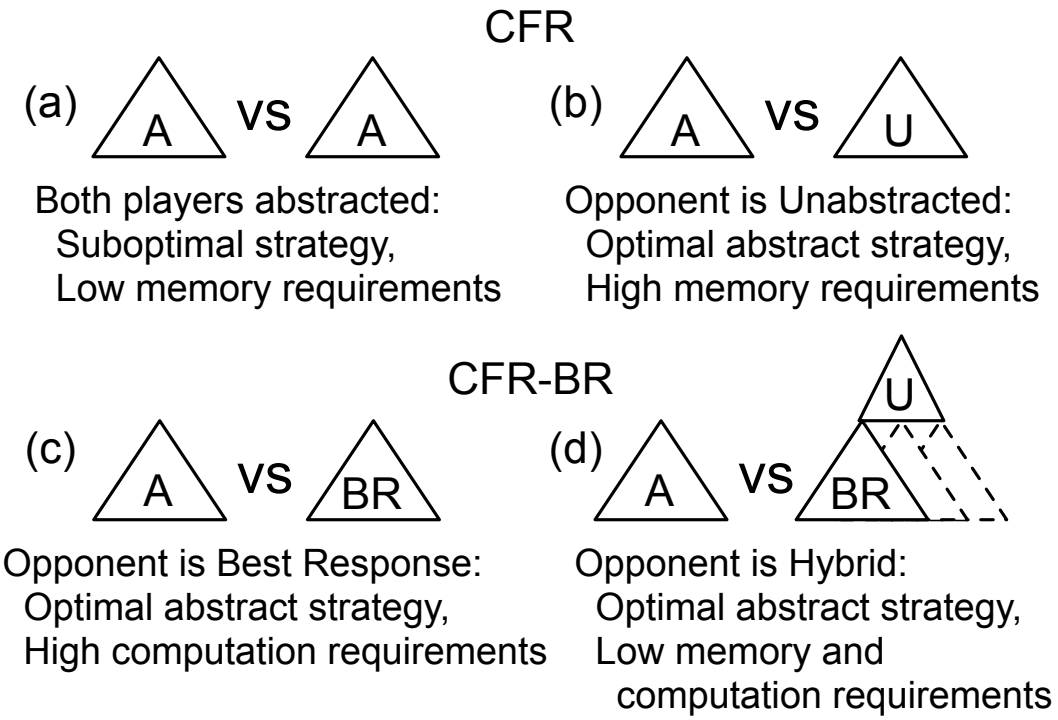


Figure 4.2: Moving from CFR to CFR-BR

In the CFR-BR algorithm, we will start with an agent that updates its strategy using CFR (a **CFR-agent**) and use a BR-agent as its opponent. The CFR-agent may use abstraction. Over a series of iterations, we will update these strategies with respect to each other. Since both of these agents are regret minimizing agents, we can prove that they converge to an equilibrium at a rate similar to the original symmetric CFR approach.

Theorem 4 *After T iterations of CFR-BR, $\bar{\sigma}_1^T$ is player 1's part of an ϵ-Nash equilibrium, with $\epsilon = \frac{\Delta|\mathcal{I}_1|\sqrt{|A_1|}}{\sqrt{T}}$.*

Proof Since player 1 is playing according to CFR, by Zinkevich et al. [110], $R_1^T \leq \epsilon$. By the folk theorem, to finish the proof it is enough to show that player 2 has no positive regret.

$$T \cdot R_2^T = \max_{\sigma_2} \left(\sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2) - \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^t) \right) \tag{4.1}$$

$$= \max_{\sigma_2} \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2) - \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^t) \tag{4.2}$$

$$= \max_{\sigma_2} \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2) - \sum_{t=1}^{T} \max_{\sigma_2'} u_2(\sigma_1^t, \sigma_2') \leq 0 \tag{4.3}$$

□

Using an unabstracted BR-agent as opposed to an unabstracted CFR-agent for the opponent has two benefits. First, its strategy will be pure, and can thus be represented more compactly than a behavioral strategy that assigns probabilities to actions. Second, we will now prove that when a CFR-agent plays against a BR-agent, the CFR-agent's sequence of strategies converges to a Nash equilibrium. Typically, it is only the *average* strategy that converges. However, since the current strategy converges with high probability, tracking the average strategy is unnecessary and only half as much memory is required for the CFR-agent. Note that the proof requires the algorithm to be stopped stochastically in order to achieve its high-probability guarantee. In practice, our stopping time is dictated by convenience and availability of computational resources, and so is expected to be sufficiently random.

Theorem 5 *If CFR-BR is stopped at an iteration T^* chosen uniformly at random from $[1, T]$, then for any $p \in (0, 1]$, with probability $(1 - p)$, $\sigma_1^{T^*}$ is player 1's part of an $\frac{\epsilon}{p}$-Nash equilibrium with ϵ defined as in Theorem 4.*

Proof As in Theorem 4, after T iterations, $R_1^T \leq \epsilon$. This gives a bound on the average observed value based on the game value v_1.

$$R_1^T = \frac{1}{T} \max_{\sigma_1} \sum_{t=1}^{T} u_1(\sigma_1, \sigma_2^t) - \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_2^t) \leq \epsilon \tag{4.4}$$

$$\therefore \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_2^t) \geq \frac{1}{T} \max_{\sigma_1} \sum_{t=1}^{T} u_1(\sigma_1, \sigma_2^t) - \epsilon \tag{4.5}$$

$$\geq \max_{\sigma_1} u_1(\sigma_1, \bar{\sigma}_2^T) - \epsilon \tag{4.6}$$

$$\geq v_1 - \epsilon \tag{4.7}$$

For all t, σ_2^t is a best response to σ_1^t, so $u_1(\sigma_1^t, \sigma_2^t) \leq v_1$. With the bounds above, this implies $u_1(\sigma_1^t, \sigma_2^t) < v_1 - \frac{\epsilon}{p}$ on no more than $\lfloor p * T \rfloor$ of the T iterations. If T^* is selected uniformly at random from $[1, T]$, there is at least a $(1 - p)$ probability that $u_1(\sigma_1^{T^*}, \sigma_2^{T^*}) \geq v_1 - \frac{\epsilon}{p}$. Because $\sigma_2^{T^*}$ is a best response to $\sigma_1^{T^*}$, this means $\sigma_1^{T^*}$ is player 1's part of an $\frac{\epsilon}{p}$-Nash equilibrium. □

CFR-BR with sampling CFR-BR still has two remaining challenges that make its use in large games intractable. First, while a best response can be stored compactly, it is still far too large to store in human-scale settings. Second, best response strategies are nontrivial to compute. Recently Johanson and colleagues demonstrated an accelerated best response technique in the poker domain that required just 76 CPU-days, and could be run in parallel in one day [53]. While previously such a computation was thought intractable, its use with CFR-BR would involve repeatedly doing this computation over a large number of iterations for convergence to a desired threshold.

However, there is an alternative. Monte-Carlo CFR (MCCFR) is a family of sampling variants of CFR in which some of the actions in a game, such as the chance events, can be sampled instead of enumerated [62]. This results in faster but less precise strategy updates for the agents, in which only subgames of the game tree are explored and updated on any one iteration. One such variant, known as Public Chance Sampled CFR (PCS), uses the fast game tree traversal from the accelerated best response technique to produce a CFR variant that efficiently traverses the game tree, updating larger portions on each iteration than were previously possible [50]. The new variant samples only public chance events while updating all possible information sets that vary in each agent's private information.

We can use a variant of PCS with CFR-BR to avoid the time and memory problems described above. On each iteration of CFR-BR, we will sample one public chance event early in the game and only update the complete subgame reachable given that outcome. This subgame includes all possible subsequent chance events after the sampled one. This divides the game tree into two parts: the **trunk** from the root to the sampled public chance event, and the **subgames** that descend from it. Unlike strategies based on regret accumulated over many iterations, portions of a best response strategy can be computed in each subgame as required and discarded afterwards. This avoids the memory problem described above, as at any one time, we only need to know the BR-agent's strategy in the trunk and the one sampled subgame for the current iteration. However, the computation problem remains, as creating the BR-agent's trunk strategy would still require us to traverse all of the possible chance events, in order to find the value of actions prior to the sampled public chance event.

To avoid this final computation problem, we replace the BR-agent with yet another regret-minimizing agent which we call a **Hybrid-agent**. This agent will maintain a strategy and regret values for the trunk of the game, and update it using Public Chance Sampled CFR. In the subgames, it will compute and follow a best response strategy to the opponent's current strategy. Together, this means that on any one iteration, we only need to compute and store one subgame of a best response, and thus require far less time and memory than a BR-agent does. We will now prove that the Hybrid-agent is a regret minimizing agent.

Definition 3 *$\tilde{\mathcal{I}}_2 \subset \mathcal{I}_2$ is a* trunk for player 2 if and only if for all $I, I' \in \mathcal{I}_2$ such that there exists $h \sqsubseteq h'$ with $h \in I$ and $h' \in I'$, if $I' \in \tilde{\mathcal{I}}_2$ then $I \in \tilde{\mathcal{I}}_2$. In other words, once player 2 leaves the trunk, she never returns to the trunk.

Theorem 6 *After T iterations of hybrid CFR-BR using a trunk $\tilde{\mathcal{I}}_2$, with probability $(1-p)$, $R_2^T \leq \epsilon = \left(1 + \frac{\sqrt{2}}{\sqrt{p}}\right) \frac{\Delta|\tilde{\mathcal{I}}_2|\sqrt{|A_2|}}{\sqrt{T}}$.*

Proof Define a partial best-response with respect to the trunk $\tilde{\mathcal{I}}_2$ as follows

$$\sigma_{2:\mathcal{I}_2 \setminus \tilde{\mathcal{I}}_2 \to \mathrm{BR}(\sigma_1)} = \underset{\sigma_2' \text{ s.t. } \sigma_2'(I)=\sigma_2(I) \forall I \in \tilde{\mathcal{I}}_2}{\operatorname{argmax}} u_2(\sigma_1, \sigma_2') \tag{4.8}$$

We can bound the regret using this partial-best response.

$$R_2^T = \frac{1}{T}\max_{\sigma_2}\sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2) - \frac{1}{T}\sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^t) \tag{4.9}$$

$$\leq \frac{1}{T}\max_{\sigma_2}\sum_{t=1}^{T} u_2\left(\sigma_1^t, \sigma_{2:\mathcal{I}_2\setminus\tilde{\mathcal{I}}_2\to \mathrm{BR}(\sigma_1^t)}\right) - \frac{1}{T}\sum_{t=1}^{T} u_2\left(\sigma_1^t, \sigma^t_{2:\mathcal{I}_2\setminus\tilde{\mathcal{I}}_2\to \mathrm{BR}(\sigma_1^t)}\right) \tag{4.10}$$

Because σ_2^t no longer has any effect outside $\tilde{\mathcal{I}}_2$, this is equivalent to doing sampled CFR on a modified game where player 2 only acts at information sets in the trunk. This means we can bound the regret by ϵ with probability $(1-p)$ by application of the MCCFR bound from Lanctot et al. [62, Theorem 5]. □

Since the Hybrid-agent is regret minimizing, it is simple to show that a CFR-agent playing against it will converge to an equilibrium using our sampling variant of CFR-BR.

Theorem 7 *For any $p \in (0, 1]$, after T iterations of hybrid CFR-BR using a trunk $\tilde{\mathcal{I}}_2$, with probability $(1-p)$, $(\bar{\sigma}_1^T, \bar{\sigma}_2^T)$ is an $(\epsilon_1 + \epsilon_2)$-Nash equilibrium profile with $\epsilon_1 = \left(1 + \frac{2}{\sqrt{p}}\right)\frac{\Delta|\mathcal{I}_1|\sqrt{|A_1|}}{\sqrt{T}}$ and $\epsilon_2 = \left(1 + \frac{2}{\sqrt{p}}\right)\frac{\Delta|\tilde{\mathcal{I}}_2|\sqrt{|A_2|}}{\sqrt{T}}$.*

Proof Because player 1 is playing according to sampled CFR, we can bound $R_1^T \leq \epsilon_1$ with probability $(1 - p/2)$ by application of the MCCFR bound [62, Theorem 5]. Theorem 6 shows that $R_2^T \leq \epsilon_2$ with probability $(1 - p/2)$. Using the union-bound, we have that both conditions hold with at least probability $(1-p)$. If both conditions hold, Theorem 2 gives us that $(\bar{\sigma}_1^T, \bar{\sigma}_2^T)$ is an $(\epsilon_1 + \epsilon_2)$-Nash equilibrium. □

Unfortunately, since the Hybrid-agent does not use a best response strategy in the trunk, only the CFR-agent's average strategy (and not the current strategy) is guaranteed to converge to a Nash equilibrium. Since the trunk is such a minuscule fraction of the tree, the current strategy might still converge (quickly) in practice. We will specifically investigate this empirically in the next section. In the remainder of the paper, we will use the name CFR-BR to refer to the variant that uses the Hybrid-agent, as this is the variant that can be practically applied to human scale problems.

4.4 Empirical Analysis

Our empirical analysis begins by exploring the correctness of our approach in a toy poker game. We then apply our technique to two-player (heads-up) limit Texas hold'em. Finally, we explore how we can use CFR-BR to answer previously

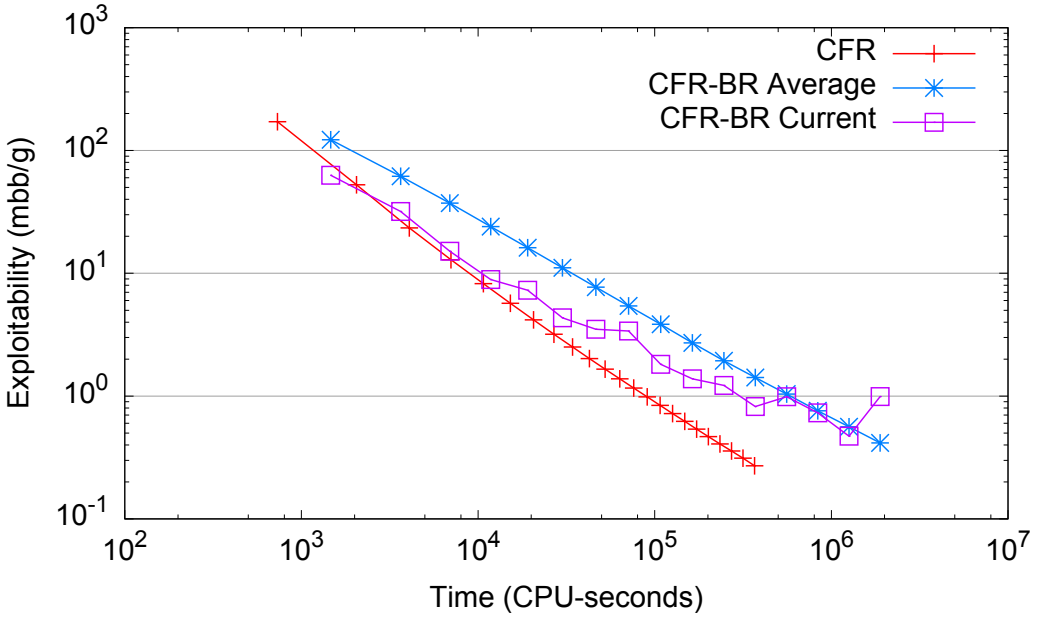


Figure 4.3: Convergence to equilibrium in unabstracted [2-4] hold'em, 94 million information sets.

unanswered questions about abstraction quality, abstraction size, and the quality of strategies in competition.

Toy Game We begin our empirical analysis of CFR-BR in the small poker game of 2-round 4-bet hold'em ([2-4] hold'em), recently introduced by Johanson et al. [50]. While we call this a "toy game", this game has 94 million canonical information sets and 2 billion game states. It is similar to the first two rounds of two-player limit Texas hold'em. A normal sized deck is used, each player is given two private cards at the start of the game, and three public cards are revealed at the start of the second round. In each round, the players may fold, call and bet as normal, with a maximum of four bets per round. At the end of the second round, the remaining player with the best five-card poker hand wins. This game is useful for our analysis because it is small enough to be solved by CFR and CFR-BR without requiring any abstraction. In addition, we can also solve this game when one or both players do use abstraction, so that we can evaluate the impact of the overfitting effect described earlier. The following [2-4] experiments were performed on a 12-core 2.66 GHz computer, using a threaded implementation of CFR and CFR-BR.

Figure 4.3 shows the convergence rate of Public Chance Sampled CFR and CFR-BR in unabstracted [2-4] hold'em on a log-log plot. In this two-round game, CFR-BR uses a "1-round trunk", and each iteration involves sampling one set of flop cards. Each series of datapoints represents the set of strategies produced by CFR or CFR-BR as it runs over time, and the y-axis indicates the exploitability of the strategy. In the computer poker community, exploitability is measured in **milli-big-blinds per game (mbb/g)**, where a milli-big-blind is one one-thousandth of a

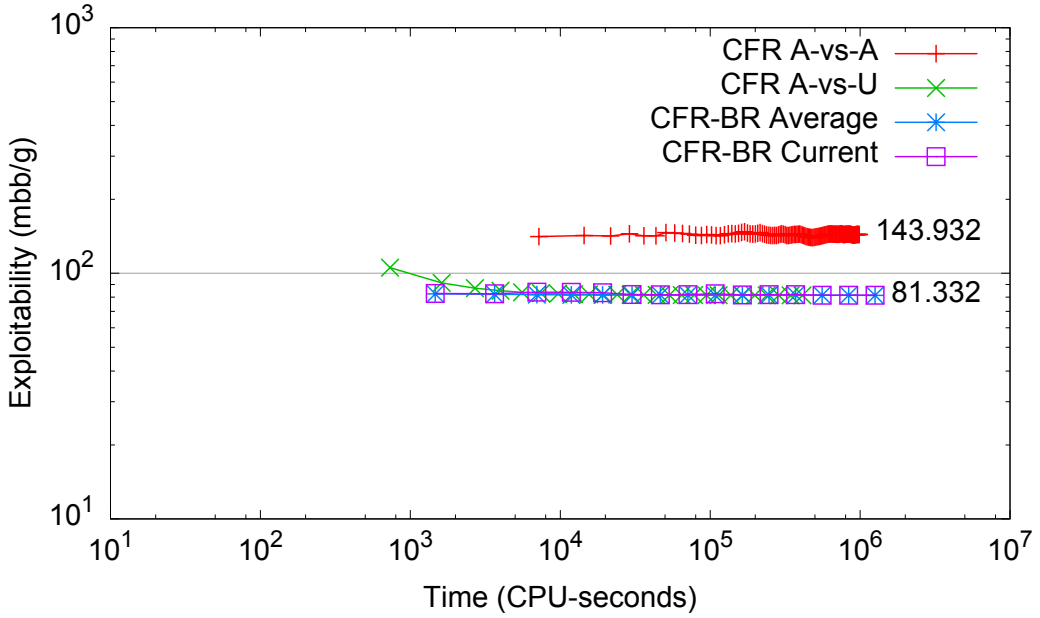


Figure 4.4: Convergence in [2-4] hold'em using a perfect recall 5-bucket abstraction, 1,790 information sets.

big blind, the ante made by one player at the start of the game. All exploitability numbers for all experiments are computed exactly using the technique in Johanson et al. [53]. From the graph, we see that CFR smoothly converges towards an optimal strategy. The CFR-BR average strategy also smoothly converges towards equilibrium, although at a slower rate than CFR. Finally, the CFR-BR current strategy also improves over time, often faster than the average strategy, although it is noisier.

In Figure 4.4, we investigate the effects of applying a simple perfect recall abstraction technique to [2-4] hold'em. When CFR solves a game where both players are abstracted (CFR A-vs-A), we see that the strategies are exploitable for 144 mbb/g in the unabstracted game. When CFR is used to create an abstracted player through games against an unabstracted opponent (CFR A-vs-U), the abstracted strategies converge to an exploitability of 81 mbb/g. This demonstrates that the abstraction is capable of representing better approximations than are found by CFR as it is typically used. With CFR-BR, both the average strategy and the current strategy converge to this same improved value.

In Figure 4.5, we perform a similar experiment where an imperfect recall abstraction is applied to [2-4] hold'em. Imperfect recall abstractions have theoretical problems (*e.g.* , the possible non-existence of Nash equilibria), but have been shown empirically to result in strong strategies when used with CFR [104; 53]. When both players are abstracted, CFR converges to an exploitability of 103 mbb/g. When only one player is abstracted, or when CFR-BR is used, the abstracted player's strategy converges to an exploitability of 25 mbb/g.

These results in [2-4] hold'em show that CFR-BR converges to the same quality of solution as using CFR with one unabstracted player, while avoiding the high

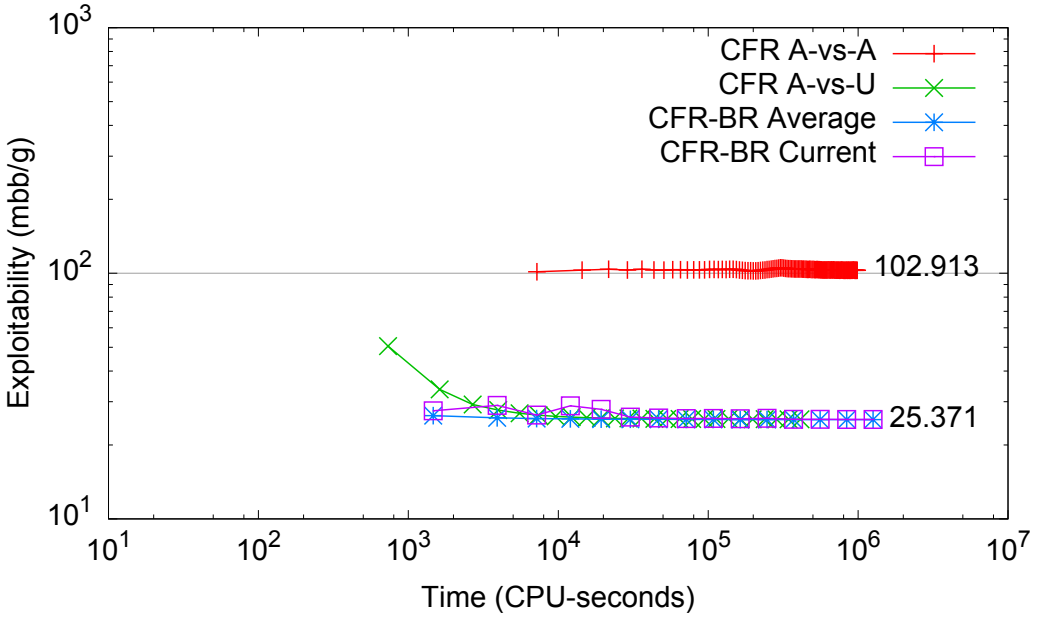


Figure 4.5: Convergence in [2-4] hold'em using an imperfect recall 570-bucket abstraction, 41k information sets.

memory cost of representing the unabstracted player's entire strategy. We also note that while the CFR-BR current strategy is not guaranteed to converge since the unabstracted Hybrid-agent uses CFR in the trunk, in practice the current strategy converges nearly as well as the average strategy. Having demonstrated these properties in a small game, we can now move to the large game of Texas hold'em in which it is intractable to use CFR with an unabstracted opponent.

Texas Hold'em We can now apply the CFR-BR technique to the large game of two-player limit Texas hold'em, one of the events in the Annual Computer Poker Competition [44]. First, we will investigate how the choice of the size of the trunk impacts the memory requirements and convergence rate. In the [2-4] hold'em results presented above, we used a "1-round trunk", where each iteration sampled the public cards revealed at the start of the second round. While the split between the trunk and the subgames could happen at any depth in the tree, in practice it is convenient to start subgames at the start of a round. In a four-round game such as Texas hold'em, there are three such convenient choices for the size of the trunk: 1-round, 2-round, or 3-round. With a 1-round trunk, each iteration involves sampling one set of public cards for the flop, and then unrolling all possible turn and river cards to create a best response strategy for this 3-round subgame. We then update the CFR-agent throughout this large subgame, and use the resulting values to perform CFR updates for both players in the trunk. Alternatively, with a 2-round trunk we will sample one set of flop and turn public cards and unroll all possible river cards. The trunk is thus larger and requires more time to update, but each subgame is smaller and updates are faster. Similarly, a 3-round trunk will sample one

	Texas hold'em RAM required		
CFR-BR Trunk	Trunk	Subgame	Total (48 cores)
1-Round	14.52 KB	1.18 GB	56.64 GB
2-Round	936.33 MB	2.74 MB	1.07 GB
3-Round	359.54 GB	6.54 KB	359.54 GB
CFR (4-round)	140.26 TB	n/a	140.26 TB

Table 4.1: Memory requirements for the CFR-BR Hybrid-agent in heads-up limit Texas hold'em

set of flop, turn and river cards, and each small subgame involves only the betting on the final round. A 4-round trunk would be equivalent to running CFR with an unabstracted opponent, as the entire game would be in the trunk.

Our choice of the size of the trunk thus allows us to trade off between the time required for the trunk and subgame updates, and the memory required to store an unabstracted CFR trunk strategy and the unabstracted best response subgame strategy. In practice, multiple threads can be used that each perform updates on different subgames simultaneously. Thus, the program as a whole requires enough memory to store one copy of the CFR player's strategy and one copy of the Hybrid-agent's trunk strategy, and each thread requires enough memory to store one pure best response subgame strategy. In Table 4.1, we present the memory required for a CFR-BR Hybrid-agent using these trunk sizes, after merging isomorphic information sets that differ only by a rotation of the cards' suits. As a 3-round trunk would require 360 gigabytes of RAM just for the Hybrid-agent, our Texas hold'em experiments will only use 1-round and 2-round trunks. Since CFR with an unabstracted opponent requires an infeasible 140 terabytes of RAM, our results will only compare CFR-BR to CFR with both players abstracted. For our experiments on Texas hold'em, a 48-core 2.2 GHz computer was used with a threaded implementation of Public Chance Sampled CFR and CFR-BR.

Figure 4.6 shows a log-log convergence graph of CFR compared to 1-round and 2-round CFR-BR's current and average strategies in a 10-bucket perfect recall abstraction. This abstraction was used to demonstrate the overfitting effect in the recent work on accelerated best response computation [53, Figure 6], and was the abstraction used by Hyperborean in the 2007 Annual Computer Poker Competition's heads-up limit instant runoff event. Due to the overfitting effect, CFR reaches an observed low point of 277 mbb/g after 2,713 seconds (130k seconds of CPU-time), but then gradually increases to an exploitability of 305 mbb/g. The 2-round trunk CFR-BR current and average strategies reach 92.638 mbb/g and 93.539 mbb/g respectively, and very little progress is being made through further computation.

Figure 4.7 demonstrates CFR and CFR-BR in a 9000-bucket imperfect recall abstraction. This abstract game is almost exactly the same size as the perfect recall abstraction presented in Figure 4.6, and was also used previously to demonstrate the overfitting effect [53, Figure 6]. In this setting, CFR reaches an observed low of 241 mbb/g within the first 3600 seconds (172k seconds of CPU-time), and then

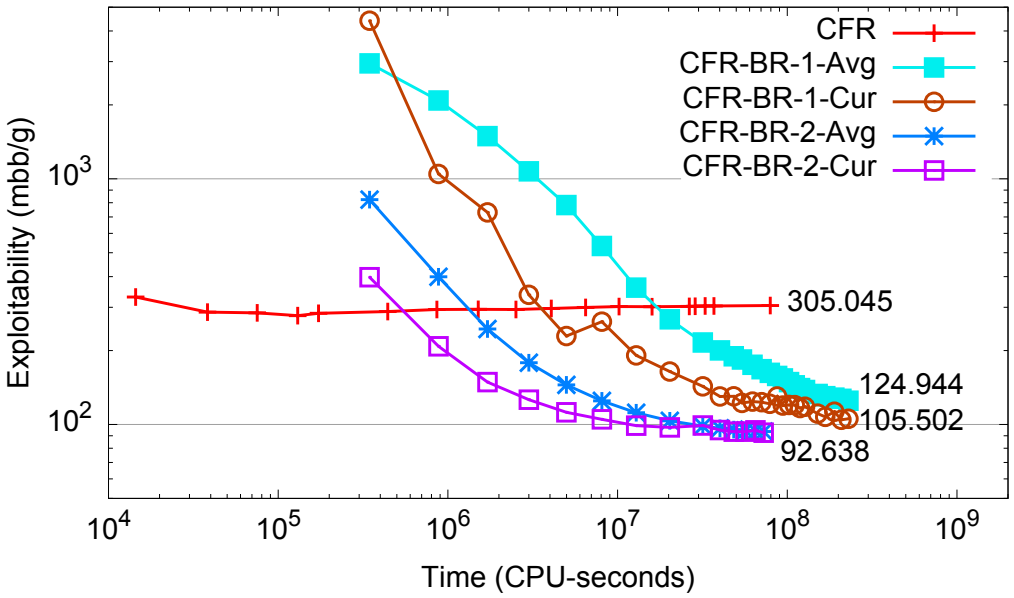


Figure 4.6: Convergence in Texas hold'em using a perfect recall 10-bucket abstraction, 57 million information sets.

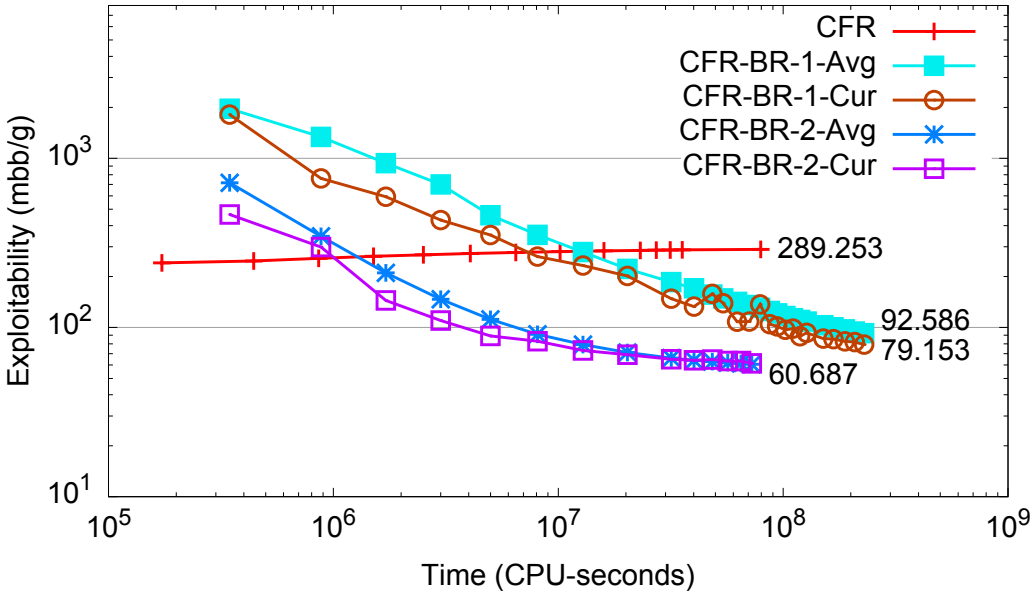


Figure 4.7: Convergence in Texas hold'em using an imperfect recall 9000-bucket abstraction, 57 million information sets.

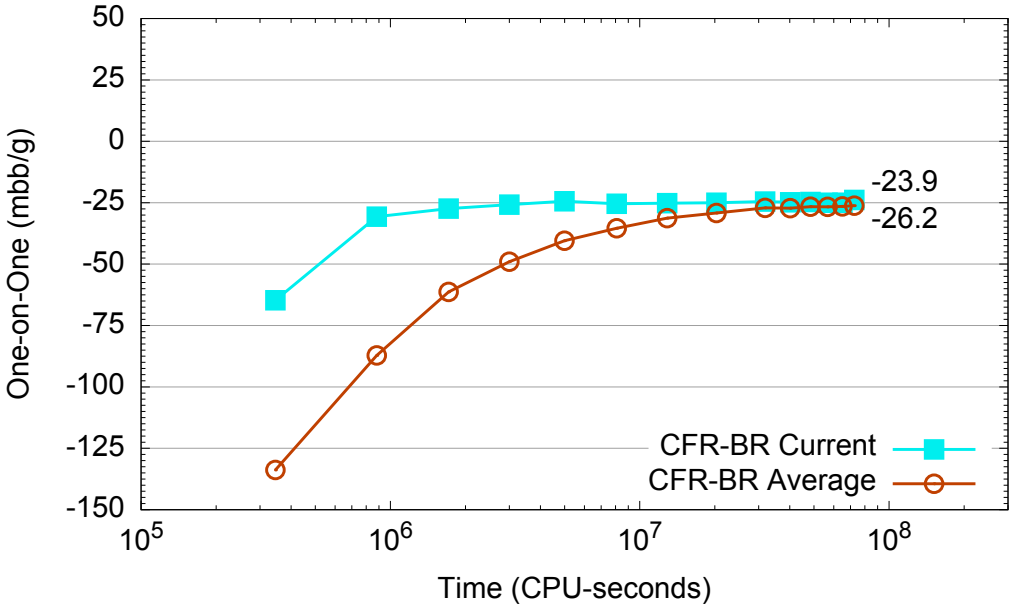
gradually increases to 289 mbb/g. The 2-round trunk CFR-BR current and average strategies reach 61.339 mbb/g and 60.687 mbb/g respectively, after which point the curves appear to have very nearly converged.

These two figures demonstrate that CFR-BR can find dramatically less exploitable strategies than is possible with CFR. The previous least exploitable known strategy for this game was Hyperborean2011.IRO, which was exploitable for 104.410 mbb/g while using an abstraction with 5.8 billion information sets, one hundred times larger than the abstractions used in Figures 4.6 and 4.7. While the 1-round and 2-round trunk strategies will converge to the same level of exploitability, we find that the 2-round trunk strategy converges significantly faster while, as shown in Table 4.1, using far less memory.

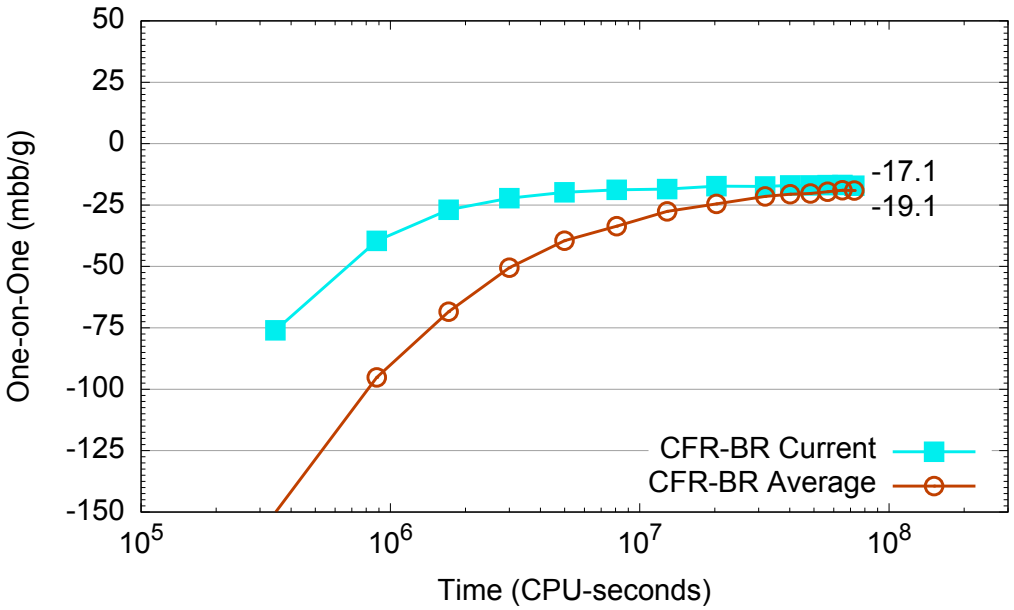
In Competition The significant drop in exploitability provided by CFR-BR is accompanied by a cost to the performance of the strategies against suboptimal opponents, such as those likely to be faced in the Annual Computer Poker Competition. When CFR is applied to an abstract game, it finds a Nash equilibrium within the abstraction and these strategies will do no worse than tie against any other strategy in the abstraction, including those generated by CFR-BR. In fact, since the CFR-BR strategies minimize their loss against an unabstracted opponent, the CFR-BR strategies will likely deviate from the abstract equilibrium in ways that incur losses against an equilibrium in the same abstraction found via CFR. Figures 4.8a and 4.8b present the in-game performance of the 2-round trunk current and average strategies from Figures 4.6 and 4.7 against the final CFR strategy from those abstractions. While the CFR-BR strategies are far less exploitable, they lose to the CFR strategies that share their abstraction.

To further investigate this effect, we can also compare the performance of CFR and CFR-BR average strategies against a CFR strategy from a much larger abstraction. In Figure 4.9, we use these same CFR and CFR-BR strategies to play games against Hyperborean2011.IRO, which uses an abstraction 100 times larger. Even though this opponent uses a much finer grained abstraction, the CFR strategies still lose less to this opponent than the CFR-BR strategies. These results underscore an observation made in the analysis of the 2010 Annual Computer Poker Competition competitors: while minimizing exploitability is a well defined goal, lower exploitability is not sufficient on its own to ensure a victory in competition against other suboptimal opponents.

Comparing Abstractions CFR-BR allows us to find optimal strategies within an abstraction. We can use this tool, then, to evaluate abstractions themselves. In the past, abstractions were typically compared by using CFR to produce strategies, and the one-on-one performance of these strategies was used to select the "strongest" abstraction. When real game best response calculations became feasible, the exploitability of the CFR strategies could instead be used to compare abstractions [53]. However, Waugh *et al.* have shown that even within one abstraction, different abstract game equilibria can have a wide range of exploitability [103, Table 3], making



(a) 10-bucket perfect recall abstraction.



(b) 9000-bucket imperfect recall abstraction.

Figure 4.8: One-on-One performance in Texas hold'em between CFR-BR strategies and the final CFR strategy with the same abstraction. Results are accurate to ± 1.2 mbb/g.

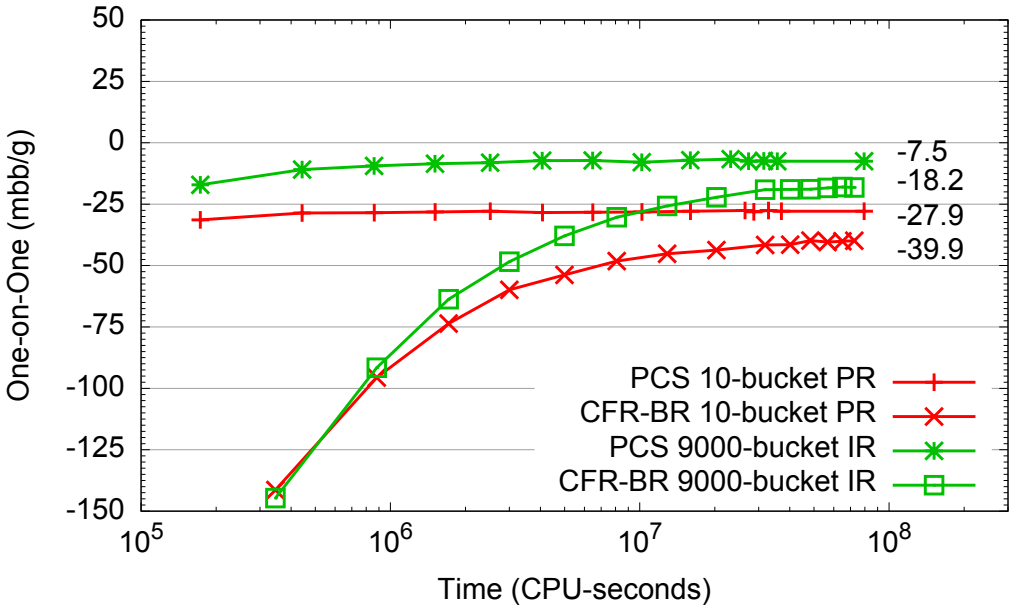


Figure 4.9: One-on-One performance in Texas hold'em between CFR-BR strategies in varying abstractions and the final CFR strategy using the Hyperborean2011.IRO abstraction. Results are accurate to ± 1.2 mbb/g.

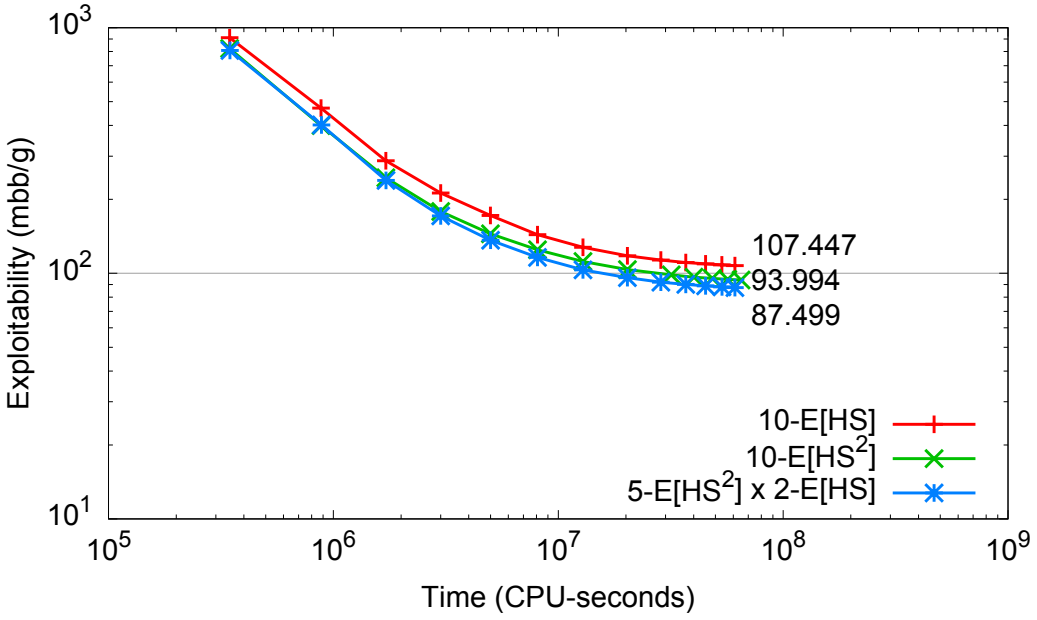


Figure 4.10: Convergence in Texas hold'em in three perfect recall 10-bucket abstractions, 57 million information sets.

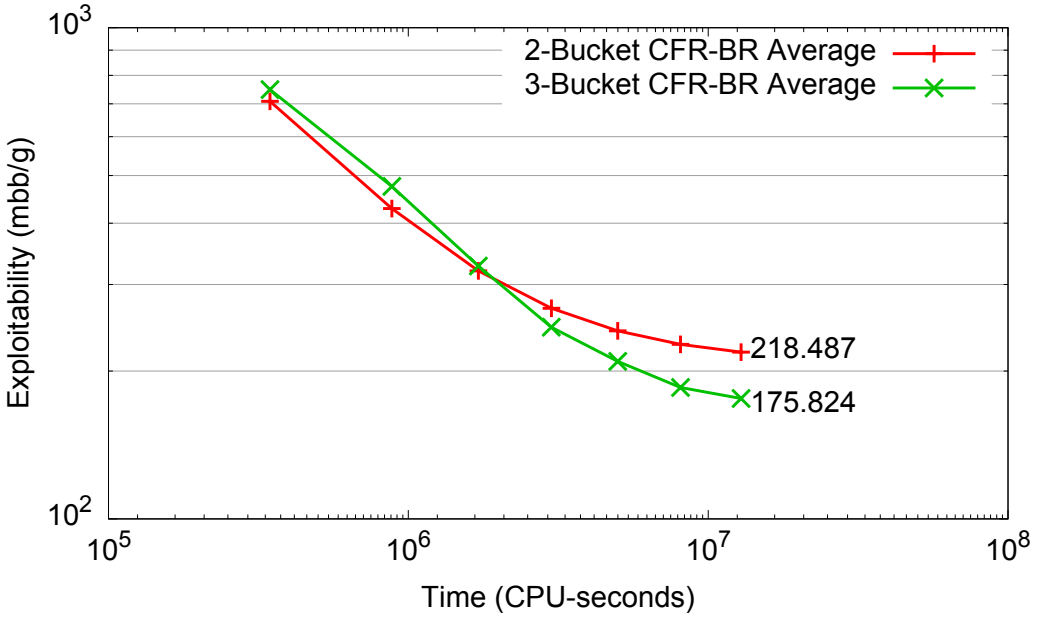


Figure 4.11: Convergence in Texas hold'em in perfect recall 2-bucket and 3-bucket abstractions, 96056 and 476934 information sets.

this approach unreliable. Since CFR-BR finds a least exploitable strategy within an abstraction, it can replace CFR in this task by directly measuring the ability of an abstraction to represent a good approximation to a Nash equilibrium.

Figure 4.10 demonstrates this abstraction comparison by applying CFR-BR to three different 10-bucket perfect recall abstractions. Each abstraction divides the set of hands into equal weight buckets according to different domain features: expected hand strength, expected hand strength squared, or a combination of both, as described in [47, Page 24]. While these abstractions are exactly the same size, we found a range of 20 mbb/g – nearly 20% – by changing the features used to create the abstraction.

Abstraction Size While abstractions can vary in the features used, they also naturally vary in size. In the 2011 Annual Computer Poker Competition entries had a hard disk limit of 30 GB, and some of the entries use large abstractions that fill this space. However, we first focus on the opposite extreme, abstractions whose strategies are so small they can fit on a single 1.44 MB floppy disk. Figure 4.11 shows the exploitability of CFR-BR strategies in extremely small 2-bucket and 3-bucket perfect recall abstractions. Despite their very coarse abstractions, the resulting strategies are exploitable for just 218.487 mbb/g and 175.824 mbb/g respectively, and are less exploitable than most of the 2010 Annual Computer Poker Competition strategies evaluated by Johanson *et al.* [53].

In Figure 4.12 we apply CFR-BR to the large, fine-grained abstraction used by Hyperborean2011.IRO in the 2011 Annual Computer Poker Competition. This abstraction has 5.8 billion information sets and in the first two rounds uses no ab-

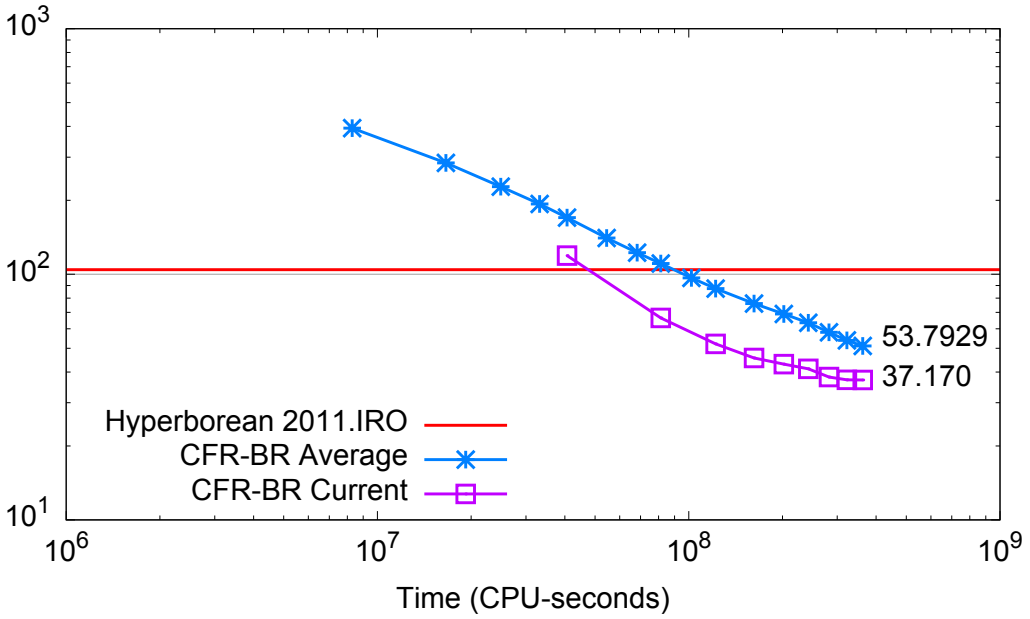


Figure 4.12: Convergence in Texas hold'em in the Hyperborean2011.IRO abstraction, 5.8 billion information sets. This figure differs slightly from Figure 12 in the original paper, as we continued running the experiment after submitting the paper. The original figure had final values of 41.199 and 63.47 for the current and average strategies respectively.

straction beyond merging isomorphic states. The turn and river rounds have 1.5 million and 840 thousand imperfect recall buckets respectively. The resulting strategy is 20 GB using only a single byte per probability. The Hyperborean2011.IRO strategy was created with CFR and was exploitable for 104.410 mbb/g, and prior to this work was the least exploitable strategy known for the game. However, by applying CFR-BR to this abstraction, the current strategy at the final datapoint is exploitable for just 37.170 mbb/g and is the new least exploitable strategy known for heads-up limit Texas hold'em poker.

4.5 Conclusion

Although there are efficient game solving algorithms for two-player, zero-sum games, many games are far too large to be tractably solved. State space abstraction techniques can be used in such cases to produce an abstract game small enough to be tractably solved; however, recent work has demonstrated that an equilibrium in an abstract game can often be far more exploitable in the unabstracted game compared to the least exploitable strategies that can be represented in the abstraction. In this work we presented CFR-BR, a new game solving algorithm that converges to one of these least exploitable abstract strategies, while avoiding the high memory cost that made such a solution previously intractable. We demonstrated the effective-

ness of our approach in the domain of two-player limit Texas hold'em, where it was used to generate far closer approximations to the unknown, optimal Nash equilibrium strategy within an abstraction than was possible using previous state-of-the-art techniques.

Acknowledgments

The authors would like to thank Marc Lanctot and the members of the Computer Poker Research Group at the University of Alberta for helpful conversations pertaining to this research. This research was supported by NSERC, Alberta Innovates Technology Futures, and the use of computing resources provided by WestGrid, Réseau Québécois de Calcul de Haute Performance, and Compute/Calcul Canada.

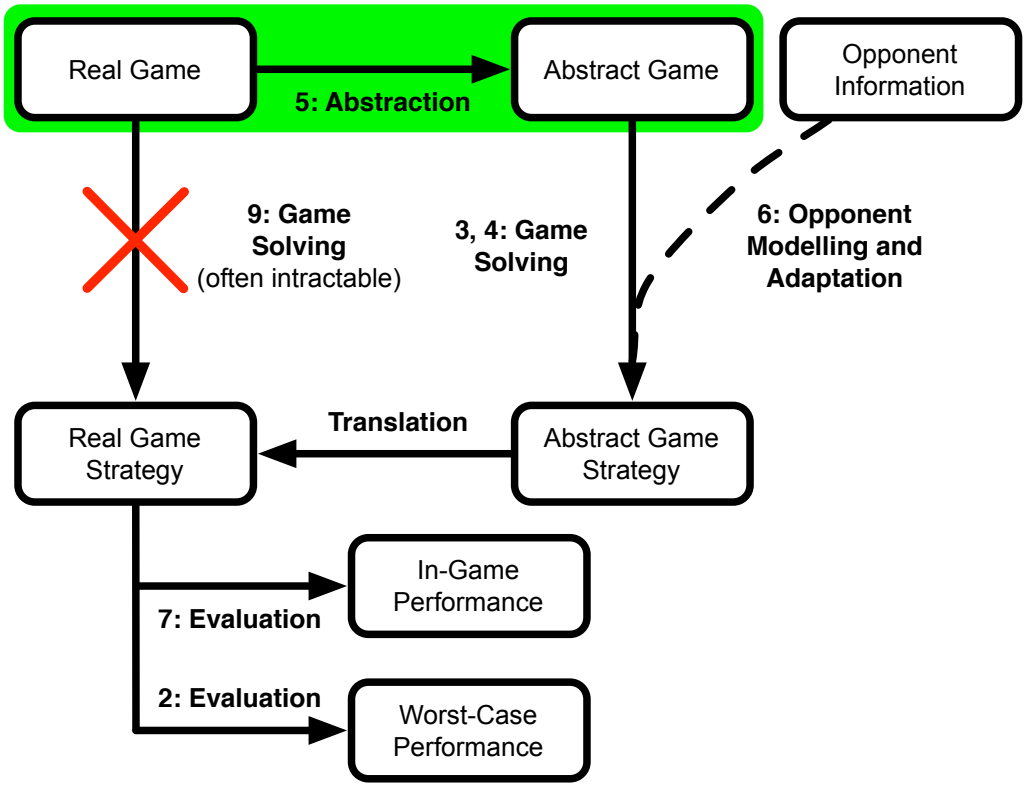
Chapter 5

State-Space Abstraction

Evaluating State-Space Abstractions in Extensive-Form Games

Michael Johanson, Neil Burch, Richard Valenzano, Michael Bowling

Presented at AAMAS 2013 [52]



Thus far, we have discussed how abstract strategies can be computed (such as by CFR, PCS, or CFR-BR) and evaluated (exploitability, in-game performance against opponents). In this chapter's paper, we will discuss how abstractions themselves can be created and evaluated. An abstraction's fidelity in modelling the real game depends both on its size and in the features and algorithms used to group real game information sets together. And although CFR-BR allows us to compute an abstraction's least exploitable strategies, higher-fidelity abstractions can represent even stronger strategies that have lower exploitability and improved in-game performance.

Prior to the invention of CFR-BR, abstraction evaluation was a difficult task. Evaluating abstractions by the exploitability of an abstract equilibrium was not always consistent, as different abstract equilibria may have large differences in real game exploitability, and abstraction pathologies and overfitting mean that the abstraction may be better than these exploitability values suggest. Comparing the in-game performance of abstract strategies is also difficult, since intransitivities may exist where strategy A beats B, B beats C, and C beats A. Using CFR-BR, we were able to bring badly needed rigour to the abstraction evaluation task by demonstrating that one abstraction can get closer to a Nash equilibrium than another. This paper also provided strong empirical evidence supporting the use of imperfect recall abstraction techniques.

This paper also presented for the first time the practical and powerful abstraction techniques that the University of Alberta had used in the Annual Computer Poker Competition from 2010 onwards. Abstraction quality is a key factor in an agent's strength, and published techniques had lagged behind those used in practice in the ACPC. This paper unveiled the details of clustering techniques, state features, and distance metrics that we had developed, and demonstrated empirically that they improved on earlier published techniques.

Author's contributions. This paper makes three contributions: the use of CFR-BR for evaluating abstractions (first proposed in our earlier paper on CFR-BR [49]), justifying the effectiveness of imperfect recall abstractions (first proposed in our earlier paper on imperfect recall [104]), and the practical and powerful abstraction technique used by the CPRG from 2009 onwards. This abstraction technique was primarily developed by myself with assistance from Valenzano, and started as a course project with Valenzano and Mengliao Wang. The use of k-means clustering as the foundation for abstraction predates this work, having been used by Martin Zinkevich in 2006 (who also experimented with earthmover's distance), and Mihai Ciucu in his ACPC agent "GGValuta" in 2008. I performed all of the experiments presented in this paper and interpreted their results. Burch contributed the software library that this work builds upon, and also contributed a "public bucket" abstraction technique that was evaluated in earlier drafts of this paper, but was not included in the final version. All four authors contributed equally to the writing and editing.

Evaluating State-Space Abstractions in Extensive-Form Games[1][2]

Michael Johanson (johanson@ualberta.ca)
Neil Burch (nburch@ualberta.ca)
Richard Valenzano (valenzan@ualberta.ca)
Michael Bowling (mbowling@ualberta.ca)

Abstract:
Efficient algorithms exist for finding optimal policies in extensive-form games. However, human-scale problems are typically so large that this computation remains infeasible with modern computing resources. State-space abstraction techniques allow for the derivation of a smaller and strategically similar abstract domain, in which an optimal strategy can be computed and then used as a suboptimal strategy in the real domain. In this paper, we consider the task of evaluating the quality of an abstraction, independent of a specific abstract strategy. In particular, we use a recent metric for abstraction quality and examine imperfect recall abstractions, in which agents "forget" previously observed information to focus the abstraction effort on more recent and relevant state information. We present experimental results in the domain of Texas hold'em poker that validate the use of distribution-aware abstractions over expectation-based approaches, demonstrate that the new metric better predicts tournament performance, and show that abstractions built using imperfect recall outperform those built using perfect recall in terms of both exploitability and one-on-one play.

5.1 Introduction

Realistic multiagent settings involve complex, sequential interactions between agents with different perspectives regarding the underlying state of the world. A general model for such settings is the extensive-form game with imperfect information. While state-of-the-art techniques for approximating Nash equilibria in extensive-form games [110; 41] have made remarkable progress [84; 50], the size of most real-world settings is beyond the capability of current solvers. For example, a common benchmark of progress is the domain of computer poker. Current solution techniques have found approximate equilibria in poker-like games with as many as 88 billion decision points [45], which is still four orders of magnitude smaller than the smallest poker game played by humans. The ubiquitous approach to handling such human-scale domains is *abstraction* [11; 89; 37], where strategically similar decision points for the players are grouped to construct an abstract game that is

[1]The contents of this chapter originally appeared at the *Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-13)*. M. Johanson, N. Burch, R. Valenzano and M. Bowling. Evaluating State-Space Abstractions in Extensive-Form Games. *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems*, 2013.

[2]The Appendix has additional results related to this paper.

tractably sized for current solution techniques. The solution of the abstract game is then employed in the original game.

While even simple abstraction techniques have been found to be empirically effective [110], their success is not guaranteed. Waugh *et al.* [103] gave surprising examples of **abstraction pathologies** where strict refinements of abstractions can result in abstract strategy equilibria that are more exploitable in the real game. While there is little theory to guide the construction of abstractions, Gilpin and Sandholm [38] presented three methods for empirically comparing abstraction methodologies: one-on-one comparison, versus-equilibrium comparison, and versus-best-response comparison. While these remained the best-practice approach for abstraction evaluation, each of these methods has conceptual drawbacks: possible intransitivities, infeasible computation, and not being well correlated with actual performance (respectively). Johanson *et al.* [49] recently presented the **CFR-BR** algorithm, which computes the best Nash approximation strategy that can be represented in a given abstraction. This represents a new, fourth method for evaluating abstraction methodologies: comparing the representation power of an abstraction by how well it can approximate an unabstracted Nash equilibrium. In this paper, we will examine the efficacy of this new approach for evaluating abstractions, and use it to evaluate several abstraction methodologies in the poker domain. We show that not only does it have many desirable conceptual properties (e.g., transitivity and computational tractability), it is also empirically well-correlated with the in-game performance of abstract game equilibria. We demonstrate all of this through a series of abstraction evaluation experiments. In particular, we repeat the Gilpin and Sandholm experiments that concluded that expectation-based abstractions are weaker than distribution-aware abstractions[3]. We also use this technique to validate the efficacy of **imperfect recall** abstractions, in which an agent forgets information known in past decisions to refine its representation of its current state. Such abstractions are empirically effective [104; 53], but previous research has not shown a conclusive advantage. Finally, we present for the first time the abstraction methodology employed by Hyperborean, one of the top competitors in the Annual Computer Poker Competition.

5.2 Background

Extensive-form games Extensive-form games are an intuitive formalism for representing the interaction between agents and their environment. These interactions are represented by a tree, in which nodes represent game states and edges represent actions taken by one of the agents, $i \in N$, or **chance**, c. The root of the tree represents the start of the interaction, and actions are taken until a leaf, i.e., **terminal node** is reached. Each terminal node $z \in Z$ assigns a utility to each player i, $u_i(z)$. In **imperfect information** games, agents may not be able to observe some of the actions taken by chance or the other agents. In the poker setting we use the terms

[3]Gilpin and Sandholm refer to their abstraction technique as being *potential-aware*, as it is distribution-aware and can also represent how quickly a hand may change over time.

private and **public** to refer to actions visible to only one agent or to all agents, although in general other types of actions are possible. Each set of game states that are indistinguishable by the acting agent is called an **information set**. When some actions are not observed, an agent perceives the game not as a tree of game states, but as a tree of information sets. A **perfect recall** game has the natural property that each agent remembers the exact sequence of its past observations and actions leading to each decision.

A **behavioral strategy** (or simply a **strategy**) for each player i, σ_i, maps each of player i's information sets to a probability distribution over the legal actions. A **strategy profile** σ is a tuple containing a strategy for each player. Let σ_{-i} refer to the strategies of player i's opponents. Given a strategy profile σ, we denote each player's **expected utility** by $u_i(\sigma)$. Given the opponents' strategies σ_{-i}, a **best response** for player i is the strategy that maximizes utility against σ_{-i}, where $b_i(\sigma_{-i})$ is the utility of the best response strategy when played against σ_{-i}. A strategy profile σ is called an **ϵ-Nash equilibrium** if $\forall i \in N, b_i(\sigma_{-i}) - u_i(\sigma_i, \sigma_{-i}) \leq \epsilon$. When $\epsilon = 0$, the profile is called a **Nash equilibrium**. In two-player repeated games where the agents alternate positions, each agent has one strategy for each position and their **exploitability** is their utility (averaged over all positions) against a worst-case adversary who, in each position, uses a best-response to the agent. In two-player zero-sum games, a Nash equilibrium has an exploitability of 0 and thus cannot lose, on expectation, to any adversary.

Poker is a canonical example of stochastic imperfect information extensive-form games. In this paper we will focus on two-player limit Texas hold'em, which is one of the variants played in the Annual Computer Poker Competition. The game begins with each player being given a **hand** of two **private cards** that only they can see or use. The players' actions are to **bet** or **call**, placing or matching wagers that their hand will be the strongest at the end of the game, or to **fold** to concede the game. This is followed by chance revealing an additional three **public cards** that both players can see and use, and an additional round of betting actions. After two additional such rounds in which one public card is revealed and the players act, the game is over and the player with the strongest hand made of their private cards and the public cards wins the wagers. Poker is a repeated game in which two agents will play a long series of such games with the overall goal of having the highest total winnings.

Counterfactual Regret Minimization Counterfactual Regret Minimization (CFR) is a state-of-the-art algorithm for **solving** extensive-form games (*i.e.* , approximating a Nash equilibrium strategy) and has been widely used in the poker domain [110; 45]. Although it is only proven to converge to a Nash equilibrium in two-player zero-sum perfect recall games, in practice it appears robust when these constraints are violated as it has been successfully applied to multi-player games [79], non-zero-sum games [53], and imperfect recall games [104]. CFR is an iterative self-play algorithm. Each player starts with an arbitrary strategy. On each iteration, the players examine every decision, and for each possible action

compare the observed value of their current policy to the value they could have achieved by making that action instead. This difference is the **regret** for playing an action, and the accumulated regret is used to determine the strategy used on the next iteration. In the limit, the average strategies used by the players will converge to a Nash equilibrium. CFR is efficient in both time and memory, requiring space which is linear in the number of actions across all information sets. While it has been applied to games with up to 8.8×10^{10} information sets [45], the computation remains intractable for domains as large as two-player limit Texas hold'em, which has 3.2×10^{14} information sets.

State-space abstraction A **state space abstraction** is a many-to-one mapping between the game's information sets and the information sets in a smaller, artificially constructed abstract game. An agent using abstraction only observes its abstract game information set, and its strategy for that information set is used for all of the real information sets mapped to it. The goal is to construct a game small enough that an optimal strategy can be found through an algorithm such as CFR, and the resulting strategy can be used to choose actions in the original game, where it is hoped to closely approximate a Nash equilibrium strategy. The success of this approach relies on both the size of the abstract game (a larger and finer-grained abstract game can lose less information) and the domain features used to decide which information sets can be mapped together.

The earliest uses of state-space abstraction in poker involved the construction of abstract chance events, called **bins** by Shi and Littman [89], **buckets** by Billings *et al.* [11], and **signals** by Gilpin and Sandholm [37], by grouping together chance events that are similar according to a metric. As the players' actions were left unabstracted, the abstract game resembles the real game except with a coarsened representation of the chance events. A common metric used in this early work is a player's **expected hand strength ($E[HS]$)**. In the final round when all public cards have been revealed, a player's **hand strength (HS)** is the probability that their hand is stronger than a uniform randomly sampled opponent hand. In the earlier rounds, expected hand strength ($E[HS]$) is the expectation of hand strength over all possible rollouts of the remaining public cards. A related metric, **expected hand strength squared ($E[HS^2]$)**, computes the expectation of the squared hand strength values, and assigns a relatively higher value to hands with the potential to improve such as flush-draws or straight-draws.[4]

These **expectation-based** metrics can be used to create abstract chance events in a number of different ways, such as bucketing based on expert-chosen ranges of $E[HS]$ values [11], bucketing based on $E[HS]$ ranges chosen so as to contain an equal number of hands (called **percentile bucketing**) [110], or by merging hands whose $E[HS]$ values differ by less than a threshold [36]. Additionally, two abstraction techniques can be **nested** by applying one and then subdividing by the other. For example, an abstraction might split the possible hands into five buckets by percentile $E[HS^2]$ and further split each into two percentile $E[HS]$ buckets, giving

[4]The rationale behind $E[HS^2]$ is discussed in detail in [47, p. 25].

ten buckets overall. The Percentile nested $E[HS^2]$ / $E[HS]$ abstraction technique has been well studied by researchers [53; 104] and was used by Hyperborean in the Annual Computer Poker Competitions from 2007 to 2009.

Gilpin *et al.* showed that expectation-based metrics have difficulty distinguishing between hands that have the potential to improve and those that do not, and that this difference is strategically important [39]. High potential hands are called **drawing hands**, which might be weak initially but have the possibility to become very strong given fortunate chance outcomes later in the game. Expectation-based abstraction techniques place these hands into buckets along with hands that have a similar $E[HS]$ values and no likelihood of improving. Abstracting these strategically distinct hands together loses information, as an abstracted agent must choose one strategy to handle both cases. While the $E[HS^2]$ metric was designed to address this fault, it was only a partial solution. Gilpin *et al.* addressed this shortcoming through a multi-pass *k*-Means abstraction technique in which the final round is clustered by $E[HS]$ and each earlier round was clustered by L_2 distance over histograms showing the probability of transitioning to the next round's buckets [39]. In later work, Gilpin and Sandholm compared these **distribution-aware** abstractions to expectation-based abstractions and found that expectation-based abstractions yielded stronger strategies in small abstractions, but are surpassed as more buckets are made available [38].

Imperfect Recall **Imperfect recall** is a relaxation of perfect recall in which agents may "forget" some of the information that it has observed. It is not typically a property of a real domain (as humans cannot be forced to forget their observations), but is instead an optional property that can be used for abstract games. When creating an imperfect recall abstraction agents can be forced to discard old observations that are no longer strategically important, thus merging the real information sets that differed in this observation. This means that an agent may be able to distinguish two information sets early in a game, but not distinguish their descendant information sets later in the game. They will perceive the game as a directed acyclic graph instead of as a tree.

An example of equal-sized perfect recall and imperfect recall abstractions in a poker-like game is shown in Figure 5.1. This game starts with a chance event, C, which deals the player a private card. Each abstraction coarsens that information and maps it to a bucket, 1 or 2, indicating that the card is in the top or bottom half of all possible cards. At the action node, A, the players take a sequence of actions, X, or Y, which is followed by a chance node at which a public card is revealed. This is where the two abstractions differ. In the perfect recall abstraction, the agent must remember its sequence of observations: 1 or 2, X or Y. The new chance information is coarsened by the perfect recall abstraction, and the agent receives one of two new buckets depending on their earlier observation. The sequences 1-1, 1-2, 2-1, and 2-2 represent different sets of chance events, and can have overlapping ranges according to metrics such as $E[HS]$: a weak hand that becomes strong may score higher than a strong hand that becomes weak. In the imperfect recall abstraction,

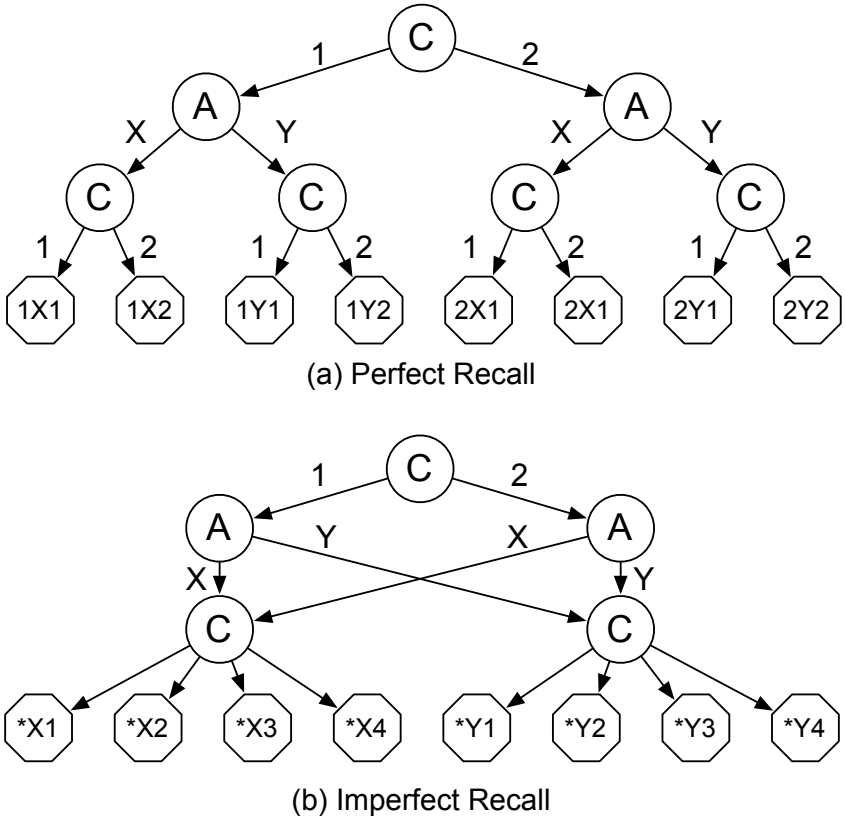


Figure 5.1: Perfect recall and imperfect recall games.

only the players' action sequence, X or Y, is remembered while the original chance bucket, 1 or 2, is forgotten. The 1X and 2X paths merge, as do 1Y and 2Y. The second chance node is coarsened to one of four buckets, 1 to 4, representing the strength of its private card combined with the public card. These four buckets can be constructed to form non-overlapping ranges of $E[HS]$. If this second chance event makes the first less significant (*i.e.* if the agent's previous strength is not very important, as is the case in poker), then the imperfect recall representation may provide more useful information.

The use of imperfect recall abstractions in the poker domain was first proposed by Waugh *et al.* [104]. As they noted, imperfect recall presents several theoretical challenges: there is no guarantee that a Nash equilibrium for an imperfect recall game can be represented as a behavioral strategy (Nash's celebrated theorem only guarantees that a mixed strategy equilibrium exists), and no proof that CFR (or other efficient algorithms) will converge towards such a strategy if one exists. Recent work by Lanctot *et al.* has shown that CFR will converge in a class of imperfect recall games; however, this class does not include the abstractions typically used in poker [61]. However, CFR remains well-defined in imperfect recall abstractions and can be used to generate abstract strategies that can be used in the real game. Waugh *et al.* [104] showed that a small improvement was possible in two-player limit Texas hold'em, as imperfect recall discarded less relevant earlier observations

and allowed new domain features to be used along with $E[HS]$.

5.3 Evaluating Abstractions

With many options available for constructing abstractions and no theory to guide these choices, progress has only been established through empirical evaluation. This involves creating abstract games, solving them, and evaluating the strategy in the real game. Gilpin and Sandholm [38] codified the possibilities for evaluating the resulting strategy, and thus evaluating the abstraction methodology itself. They described three approaches: in-game performance against other agents, in-game performance against an unabstracted Nash equilibrium, and exploitability in the real game.

While these evaluation methods measure qualities we want, each involves a potentially serious drawback. In one-on-one play, it is possible to find intransitivities where strategy *A* defeats *B*, which defeats *C*, which defeats *A*. A weaker form of intransitivity occurs when *A* defeats *B*, but *B* defeats *C* by more than *A* defeats *C*. It is not clear what to conclude in such cases. In one-on-one play against a Nash equilibrium, many strategies of varying exploitability may tie. Even more problematic is that generating an unabstracted equilibrium strategy in human-scale domains is intractable. Finally, while measuring the exploitability of abstract strategies directly addresses the goal of approximating a Nash equilibrium, recent research has shown that abstract game equilibria may not be the abstract strategies with the lowest real game exploitability [103; 53]. In addition, both Waugh *et al.* [101, p.30 and p.52] (in a toy game) and Johanson *et al.* [53] (in Texas Hold'em), found that exploitability does not correlate well with one-on-one performance.

Johanson *et al.* recently presented CFR-BR: a CFR variant that, in perfect recall abstractions, converges towards an abstract strategy with the lowest real game exploitability [49]. These strategies are not abstract game equilibria, as are found by CFR, but instead are the closest approximations to a real game equilibrium that can be represented within an abstraction. In practice, the exploitability of these CFR-BR strategies is as little as $\frac{1}{3}$ of those found via CFR. While CFR-BR's convergence is only proven for perfect recall abstractions, in practice the same degree of improvement is shown in imperfect recall games. CFR-BR requires repeated traversals of the real game tree, and may not be tractable in large domains where abstraction enables CFR. However, calculating a strategy's exploitability also requires a real game tree traversal (although an efficient traversal may be possible [53]), and in such large games one-on-one performance may remain the only viable evaluation.

Johanson *et al.* also demonstrated that CFR-BR could be used for evaluating abstractions by measuring the closest approximation to a Nash equilibrium that can be represented by the abstraction [49]. In this paper, we will broadly apply the CFR-BR technique for the first time to compare new and existing abstraction techniques. Our experiments will evaluate two abstraction choices that have been raised by recent publications: the effectiveness of expectation-based as opposed to

distribution-aware abstractions, as proposed by Gilpin and Sandholm [39], and the use of perfect recall as opposed to imperfect recall, as proposed by Waugh *et al.* [104]. We will also present for the first time the abstraction technique and distance metrics used by the Hyperborean agent in the Annual Computer Poker Competition since 2010.

5.4 Abstraction as Clustering

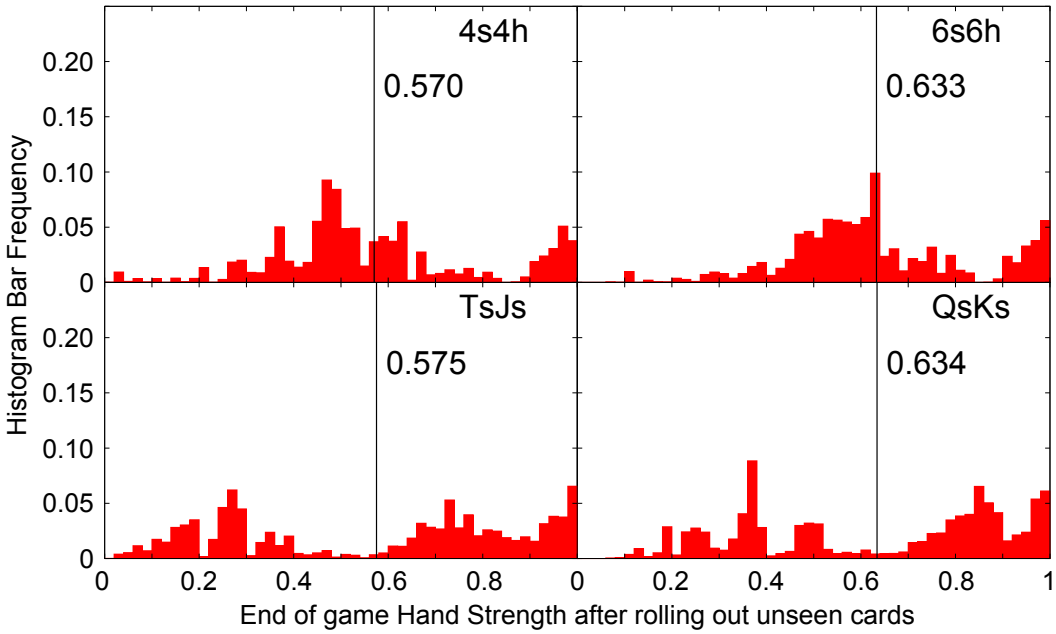
To eliminate the need for direct human expert knowledge when creating an abstraction, the abstraction generation problem will be considered as a clustering problem. Given a target number of clusters (*i.e.* buckets) k and a distance function between information sets, a clustering algorithm can be used to partition the real information sets into the buckets that form the information sets of the abstract game. Using a clustering algorithm allows the abstraction designer to focus on two aspects of the task: designing a distance metric that represents the strategic similarity of two information sets, and choosing the total number of clusters on each round, k_i, so that the total number of information sets in the resulting abstract game is small enough to solve.

In practice, the number of information sets to be clustered can be very large, making the use of many clustering algorithms computationally intractable. In the poker domain, for example, the final round of Texas hold'em has 2,428,287,420 canonical combinations of public and private cards to be grouped into between one thousand (a small abstraction) and one million (a large abstraction) clusters or more. To make this large clustering problem tractable, we use a *k*-Means implementation that uses the triangle inequality to reduce the number of distance function calls [28]. Multiple restarts and the *k*-Means ++ initialization [4] are also used to improve the quality of the clustering.

As in previous work in the limit Texas hold'em poker domain, the abstractions used in our experiments will only merge information sets on the basis of having similar chance events. This approach leaves the players' actions unabstracted and reduces the abstraction generation task to that of finding clusters of similar private and public cards. In the remainder of this section, we will present two new distance metrics for the poker domain that capture strategic similarities that were not handled by earlier expectation-based approaches, and describe how imperfect recall can be used to reallocate the distribution of buckets throughout the game.

Hand Strength Distributions In Section 5.2, we described the expected hand strength metric. In the final round of the game, hand strength measures the probability of winning against a randomly sampled opponent hand, given the public cards. Earlier in the game, $E[HS]$ measures the expectation of hand strength over all possibilities for the remaining public cards. Thus, $E[HS]$ summarizes the *distribution* over possible end-game strengths into a single expected value.

As noted by Gilpin and Sandholm [38], this single value is unable to distinguish hands with differing potential to improve. Consider Figure 5.2(top), which shows



	Earth mover's Distance				E[HS] Distance			
	4s4h	6s6h	TsJs	QsKs	4s4h	6s6h	TsJs	QsKs
4s4h		3.101	6.212	5.143		0.063	0.005	0.064
6s6h	3.101		6.462	5.286	0.063		0.015	0.001
TsJs	6.212	6.462		3.103	0.005	0.015		0.059
QsKs	5.143	5.286	3.103		0.064	0.001	0.059	

Figure 5.2: (top) Hand Strength histograms for four poker hands at the start of the game. (bottom) Earth mover's and E[HS] distances.

the distributions over the final round hand strength of four Texas hold'em poker hands in the first round of the game. Each distribution is discretized into a histogram with values ranging from 0 (a guaranteed loss) to 1 (a guaranteed win). The height of each bar indicates the probability of the remaining public cards resulting in that hand strength, and the vertical black line and label shows $E[HS]$.

Note that the top and bottom histograms have different distribution shapes: $4\spadesuit 4\heartsuit$ and $6\spadesuit 6\heartsuit$ have most of their weight near their $E[HS]$ values, while $T\spadesuit J\spadesuit$ and $Q\spadesuit K\spadesuit$ have almost no weight near $E[HS]$ as the unrevealed cards will make this hand either strong or weak. This difference is an indication that the top and bottom rows are strategically distinct: the bottom row has high potential, while the top row does not. However, when comparing the columns of hands we find almost identical $E[HS]$ values. As such, expectation-based approaches would merge within each column, whereas merging along each row may be better.

This suggests the use of a *distribution-aware* similarity metric such as earth mover's distance [105] to compare two hand strength distributions. Earth mover's distance measures the minimum work required to change one distribution into another by moving probability mass. In one-dimensional discrete distributions such as these hand strength distributions, it can be efficiently computed with a single pass over the histogram bars. Unlike alternative distance metrics such as L_2 or Kolmogorov-Smirnov, earth mover's distance measures not only the difference in probability mass, but also how far that mass was moved. In Figure 5.2(bottom), the earth mover's distance and difference in $E[HS]$ for four hands are listed. In partitioning these four hands into two clusters, earth mover's distance would merge the rows (similar distribution shapes) while $E[HS]$ would merge the columns (similar expected values).

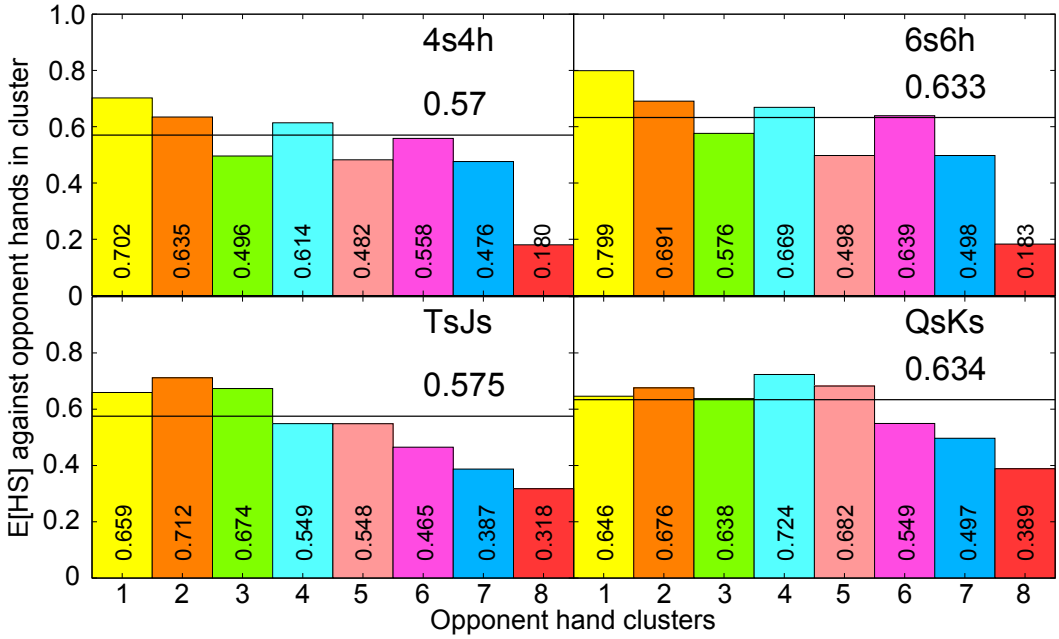
In Texas hold'em poker, hand strength histograms can be precomputed for every combination of private and public cards in the first three rounds, and earth mover's distance provides a candidate distance function for comparing them. After all of the public cards are revealed in the final round, each histogram would be a single impulse at the corresponding hand strength value, and earth mover's distance and the difference in hand strength values would be equivalent.

Opponent Cluster Hand Strength Our second new distance metric addresses a different aspect of $E[HS]$. The 'hand strength' component of $E[HS]$ measures the probability of winning against a uniform randomly sampled opponent hand at the end of the game, and this provides one summary feature. However, we can also consider our probability of winning against multiple subsets or distributions of possible opponent hands, and thereby generate additional features. While any number of overlapping or non-overlapping subsets could be used, in this work we will partition the 169 starting hands into eight non-overlapping subsets, which we call **opponent clusters**[5]. These were formed by clustering the hands using the earth mover's distance metric on the first round, and are presented in Table 5.1.

[5]Our use of these eight clusters was an engineering decision to limit the memory required for the precomputed tables; other choices may be even more effective.

		Suited												
		2	3	4	5	6	7	8	9	T	J	Q	K	A
Unsuited	2	4	1	1	1	1	1	2	2	2	4	4	4	6
	3	1	6	1	1	1	1	2	2	3	4	4	6	6
	4	1	1	6	1	1	2	2	2	3	4	4	6	6
	5	1	1	1	6	3	3	3	3	3	4	4	6	6
	6	1	1	1	1	7	3	3	3	3	4	5	6	6
	7	1	1	1	2	3	7	3	3	5	5	5	6	7
	8	1	1	2	2	3	3	8	3	5	5	5	6	7
	9	2	2	2	2	3	3	3	8	5	5	5	7	7
	T	2	2	2	2	3	3	3	5	8	5	7	7	7
	J	2	2	4	4	4	4	5	5	5	8	7	7	7
	Q	4	4	4	4	4	4	5	5	5	5	8	7	7
	K	4	4	4	6	6	6	6	6	7	7	7	8	7
	A	6	6	6	6	6	6	6	7	7	7	7	7	8

Table 5.1: Eight hand clusters used for the OCHS features.



	OCHS L_2 Distance				E[HS] Distance			
	4s4h	6s6h	TsJs	QsKs	4s4h	6s6h	TsJs	QsKs
4s4h		0.066	0.095	0.104		0.063	0.005	0.064
6s6h	0.066		0.117	0.105	0.063		0.015	0.001
TsJs	0.095	0.117		0.098	0.005	0.015		0.059
QsKs	0.104	0.105	0.098		0.064	0.001	0.059	

Figure 5.3: (top) OCHS values for four poker hands at the start of the game. (bottom) OCHS L_2 and E[HS] distances.

	# Action Sequences		PR 10-10-10-10		IR 10-100-1,000-10,000		IR 169-9,000-9,000-9,000	
Round	Inside	Continuing	# Buckets	# Infosets	# Buckets	# Infosets	# Buckets	# Infosets
1	8	7	10	80	10	80	169	1,352
2	7*10	7*9	10*10	700	100	700	9,000	630,000
3	7*9*10	7*9*9	10*10*10	630,000	1,000	630,000	9,000	5,670,000
4	7*9*9*10		10*10*10*10	56,700,000	10,000	56,700,000	9,000	51,030,000
Total				57,330,780		57,330,780		57,331,352

Table 5.2: Computing the number of information sets in three nearly equally sized Texas hold'em abstractions.

Instead of using a single $E[HS]$ value, we will now compute eight values measuring the hand strength against hands drawn uniform randomly from each opponent cluster. For example, the eighth **Opponent Cluster Hand Strength (OCHS)** feature measures the probability of winning against an opponent hand sampled from the set of top pairs. For each game round, we can precompute a vector of OCHS features to describe a hand's strength. The L_2 distance between two vectors is then used as a distance metric. Figure 5.3 shows an example with the four first-round hands from Table 5.2 and the L_2 distances between their vectors. OCHS provides a richer representation of strength than $E[HS]$, which can itself be derived from the vector.

Perfect and Imperfect Recall We will now describe how clustering can be used to form abstractions with perfect and imperfect recall. A perfect recall abstraction is created hierarchically by solving many small clustering problems. To start, the first round of the game is clustered into k_1 clusters. In the second round, perfect recall requires that information sets may only be clustered together if they share the same sequence of observations. This means that we must solve k_1 independent clustering problems, each of which only includes those chance events that are descendents of chance events clustered together in the first round. Although each of these independent clustering problems could assign a different number of clusters, in our experiments we use the same constant k_2 for each. The hierarchical abstraction generation continues until the final round in which we have to solve $k_1 \cdot \ldots \cdot k_{n-1}$ clustering problems, into k_n clusters each, for a total of $k_1 \cdot \ldots \cdot k_n$ clusters in the final round.

When creating an imperfect recall abstraction, we simply cluster all of the chance events without considering their predecessors' clusters on earlier rounds.[6] Solving one large clustering problem is more computationally difficult than solving many small ones. However, the larger number of clusters may allow for a more accurate clustering, as there will not be a need for clusters with similar features that differ only by their history.

The key constraint when making an abstraction is not the number of buckets

[6]In general, the previous rounds' clusters or features *can* be considered when forming clusters, and perfect recall simply enforces this as a constraint. In earlier work [104], we created imperfect recall abstractions that formed clusters from hands with similar, but not strictly matching, earlier clusters or features. This form of abstraction may still be useful, particularly for representing earlier public knowledge, and warrants further study. At this time, when clustering on private features such as PHS, DIST, and OCHS, we have found little benefit to considering the earlier clusters or features, and instead focus solely on the current information.

either in each round or overall, but the total number of information sets in the resulting game, as this determines the memory required to solve it. In imperfect recall abstractions it is possible to change the distribution of buckets throughout the game, dramatically increasing the number of buckets in early rounds, without changing the overall number of information sets. We demonstrate this effect in Table 5.2. The 'Action Sequences' columns describe only the players' actions and not the chance events, and shows the number of action sequences leading to a choice inside the round and continuing to the next round. The next three sections describe nearly equally sized abstractions. PR-10-10-10-10 uses perfect recall, while IR-10-100-1,000-10,000 and IR-169-9,000-9,000-9,000 use imperfect recall. For each abstraction, the table lists the number of buckets and the number of information sets (buckets times decision points) in the abstraction in that round. The final row shows the total number of information sets.

The PR-10-10-10-10 and IR-10-100-1,000-10,000 abstract games are exactly the same size and use the same total number of buckets on each round: either through multiple small perfect recall clusterings, or in one large imperfect recall clustering. The IR-169-9,000-9,000-9,000 abstraction changes the distribution of buckets, shrinking the final round to 9,000 buckets and removing 5.67 million final round information sets. Due to the multiplying effect of the number of action sequences that reach the final round, removing one fourth-round bucket allows for the addition of 9 third-round buckets, 81 second-round buckets, or 567 first-round buckets. In this way, we can decrease the number of fourth-round buckets by 10% to get an abstraction that is lossless in the first round (*i.e.* it has 169 buckets) and has 9,000 buckets in the second and third rounds. Note that this type of redistribution is not possible when using perfect recall, as the larger number of buckets early in the game need to be remembered until the final round: having 169 buckets in the first round would allow only four buckets on each subsequent round.

5.5 Results

We can now begin our empirical investigation of abstraction techniques, using the domain of two-player limit Texas hold'em poker. In this paper, we have described three abstraction techniques that are applicable to the first three rounds: Percentile Hand Strength (PHS), *k*-Means earth mover (KE), and *k*-Means OCHS (KO). We have two choices of abstraction techniques to use on the final round: Percentile Hand Strength (PHS) and *k*-Means OCHS (KO). Each combination of an early-game and end-game technique can be used to form a different abstraction. Additionally, we can consider abstractions that use Perfect Recall (PR) and Imperfect Recall (IR), resulting in $2 \times 3 \times 2 = 12$ abstractions. An abstraction (or agent) named IR-KE-KO uses imperfect recall, *k*-Means earth mover to abstract the first three rounds, and *k*-Means OCHS to abstract the final round. Each abstraction will be of the sizes listed in Table 5.2: either Perfect Recall 10-10-10-10, or Imperfect Recall 169-9000-9000-9000 with a lossless first-round abstraction. In the first three rounds, PHS abstractions will use nesting to partition first by $E[HS^2]$ and then by

$E[HS]$. Perfect recall PHS will use $5 \times 2 = 10$ buckets and imperfect recall PHS will use $150 \times 60 = 9000$ buckets. On the final round $E[HS^2]$ ranks hands in the same order as $E[HS]$, and so PHS uses a single partition into 10 or 9000 buckets.

We begin our evaluation of these abstraction styles and distance metrics with the first evaluation technique suggested by Gilpin and Sandholm: one-on-one performance between abstract game Nash equilibrium strategies [38]. For each abstraction, a parallel implementation of the Public Chance Sampled CFR algorithm (PCS) [50] was run for 4 days on a 48-core 2.2 GHz AMD computer[7]. Each pair of strategies was then played against each other for 10 million hands of duplicate poker to obtain statistically significant results with a 95% confidence interval of 1.1 mbb/g. The crosstable of this match is shown in Table 5.3. We find that every imperfect recall agent, regardless of abstraction technique, outperformed every perfect recall agent. Comparing each imperfect recall agent against its perfect recall equivalent (*i.e.* , IR-KE-KO to PR-KE-KO) we find that the imperfect recall agent also had a higher expected value against each opponent. Overall, the IR-KE-KO agent was undefeated and additionally scored the highest against each adversary. Ranked by average performance, the IR-KO-KO and IR-KE-PHS agents placed second and third.

Gilpin and Sandholm's third abstraction evaluation technique is to calculate the real game exploitability of abstract game Nash equilibrium strategies. In the CFR column of Table 5.4, we present the exploitability of the same CFR strategies used in the one-on-one crosstable. Note that the results are inconsistent: neither perfect recall or imperfect recall shows a clear advantage. Notably, the two KE-KO strategies are almost exactly tied, despite the fact that IR-KE-KO was considerably stronger in the crosstable. As described earlier, recent work by Waugh *et al.* [103] and Johanson *et al.* [53] has shown that abstract game Nash equilibria are rarely the least exploitable strategies representable in an abstraction, making this method of evaluating abstractions inconclusive.

The recently developed CFR-BR algorithm provides a more reliable metric [49]. In each abstraction, a parallel implementation of CFR-BR was run for 8 days on the same computer used to generate the CFR strategies[8]. The exploitability of these CFR-BR strategies is presented in Table 5.4, and the results are much more consistent with the one-on-one performance presented in Table 5.3. IR-KE-KO, IR-KO-KO, and IR-KE-PHS are once again ranked first, second and third. With the exception of PHS-PHS, the imperfect recall agents are also less exploitable than their perfect recall equivalents. Johanson *et al.* note that CFR-BR strategies tend to lose slightly when played against their more exploitable PCS equivalents [49, Fig. 8], and so the CFR strategies' one-on-one performance is of more interest. The outcomes of playing the CFR-BR agents against each other are very similar to those of the CFR agents in Table 5.3.

In Table 5.2, we showed that imperfect recall allows us to decrease the number

[7]Johanson *et al.* found that applying PCS to 10-bucket PR-PHS-PHS for 10^5 seconds was sufficient for near convergence [50, Figure 3c].

[8]Johanson *et al.* found that this time, 3.3×10^7 seconds, was sufficient for near convergence using PR-PHS-PHS and IR-KE-KO [49, Figures 6 and 7].

		Perfect Recall						Imperfect Recall						Mean
		PHS-PHS	PHS-KO	KE-PHS	KE-KO	KO-PHS	KO-KO	PHS-PHS	PHS-KO	KE-PHS	KE-KO	KO-PHS	KO-KO	
PR	PHS-PHS		-0.9	-1.9	-2.7	14.3	14.5	-8.8	-14.8	-21.1	-25.7	-19.1	-22.4	-8.1
	PHS-KO	0.9		-0.3	-2.3	14.2	13.5	-8.7	-14.7	-21.0	-26.3	-17.6	-22.2	-7.7
	KE-PHS	1.9	0.3		-2.4	15.2	14.5	-6.5	-13.0	-18.0	-24.2	-16.1	-21.6	-6.4
	KE-KO	2.7	2.3	2.4		16.9	16.3	-6.6	-12.5	-18.6	-24.8	-15.8	-21.0	-5.3
	KO-PHS	-14.3	-14.2	-15.2	-16.9		-1.69	-25.7	-30.3	-34.9	-39.9	-32.6	-37.3	-23.9
	KO-KO	-14.5	-13.5	-14.5	-16.3	1.69		-25.1	-30.2	-34.8	-39.6	-31.8	-37.2	-23.3
IR	PHS-PHS	8.8	8.7	6.5	6.6	25.7	25.1		-5.5	-15.5	-18.9	-10.8	-15.0	1.4
	PHS-KO	14.8	14.7	13.0	12.5	30.3	30.2	5.5		-9.6	-14.6	-6.0	-10.4	7.3
	KE-PHS	21.1	21.0	18.0	18.6	34.9	34.8	15.5	9.6		-6.7	2.6	-3.8	15.1
	KE-KO	25.7	26.3	24.2	24.8	39.9	39.6	18.9	14.6	6.7		9.3	2.8	21.2
	KO-PHS	19.1	17.6	16.1	15.8	32.6	31.8	10.8	6.0	-2.6	-9.3		-7.2	11.9
	KO-KO	22.4	22.2	21.6	21.0	37.3	37.2	15.0	10.4	3.8	-2.8	7.2		17.8

Table 5.3: Average performance in games between abstract strategies generated by Public Chance Sampled CFR. Results are in milli-big-blinds/game (mbb/g) over a 10 million hand duplicate match with a 95% confidence interval of 1.1 mbb/g. An extended version of this table is available in the appendix as Table 11.3.

	CFR		CFR-BR	
	PR	IR	PR	IR
PHS-PHS	288.942	358.188	89.632	94.841
PHS-KO	289.152	318.197	90.371	85.275
KE-PHS	281.63	339.872	90.720	80.557
KE-KO	282.856	282.395	84.039	64.820
KO-PHS	335.902	355.881	105.389	88.546
KO-KO	330.319	291.574	105.523	73.091

Table 5.4: Exploitability of CFR-BR and CFR strategies. Results are measured in milli-big-blinds/game and are exact. An extended version of this table is available in the appendix as Table 11.4.

Abstraction	CFR-BR Exploitability
PR KE-KO 10-10-10-10	84.039
IR KE-KO 10-100-1000-10000	89.7975
IR KE-KO 169-9000-9000-9000	64.820

Table 5.5: Effect of redistributing buckets in an abstraction.

of buckets in later rounds of the game in return for many more buckets in earlier rounds, without increasing the size of the game. In Table 5.5, we revisit this decision and also consider an IR KE-KO 10-100-1,000-10,000 abstraction. We find that the imperfect recall agent is more exploitable than its perfect recall equivalent, while the redistributed agent shows a significant decrease.

We can also measure the exploitability of CFR-BR strategies as a response to abstraction size, to investigate if these abstraction techniques improve at different rates. For this experiment, we consider five sizes of four abstractions: PR-PHS-PHS and IR-PHS-PHS, PR-KE-KO and IR-KE-KO. The perfect recall abstractions branch to 5, 6, 8, 10 and 12 buckets on each round, and the imperfect recall abstractions have a lossless first round and 570, 1175, 3700, 9000 and 18630 buckets on later rounds.

The CFR-BR exploitability results for these abstractions are presented in Figure 5.4 as a log-log plot. Comparing the slope of each curve, we find that IR-KE-KO and PR-KE-KO are steeper than PR-PHS-PHS and IR-PHS-PHS, indicating that their advantage increases with the abstraction size. The combination of abstraction techniques presented in this paper, imperfect recall with redistribution and the KE and KO techniques, is less exploitable at all tested abstraction sizes.

5.6 Discussion

Recent research towards state-space abstraction in the poker domain has raised two issues: the effectiveness of distribution-aware as compared to expectation-based approaches (as described by Gilpin and Sandholm [38]) and the practical uses of imperfect recall (as described by Waugh *et al.* [104]). The discovery that the ex-

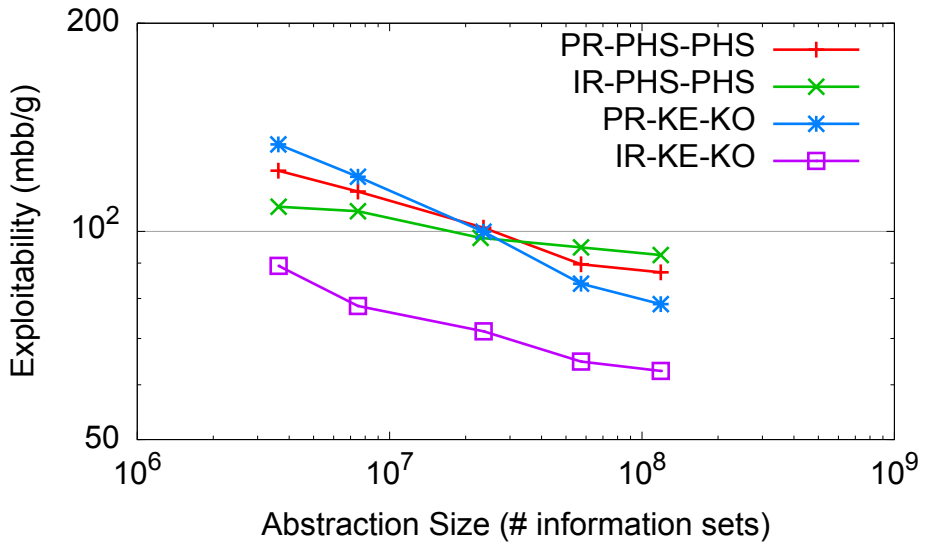


Figure 5.4: Exploitability of CFR-BR strategies in four abstractions as the abstraction size is varied. An extended version of this figure is available in the appendix as Figure 11.3.

ploitability of abstract game Nash equilibrium strategies was not an accurate measure of an abstraction's ability to represent a real Nash equilibrium has left these issues unresolved. Our goal in these experiments was to use the recently developed CFR-BR technique to survey these abstraction choices and evaluate them more precisely.

Gilpin and Sandholm's investigation showed that while agents in expectation-based abstractions are more effective in small abstract games, the distribution-aware agents match and surpass them as the number of buckets is increased. Figure 5.4 shows that our experiment matches their result: the steeper slope of the PR-KE-KO line as compared to PR-PHS-PHS shows that the distribution-aware metric makes better use of the available buckets as the abstraction size increases. In addition, the one-on-one crosstable in Table 5.3 shows that the distribution-aware agents using the k-Means earth mover's abstractions outperformed the expectation-based agents.

We now turn to imperfect recall. In one-on-one performance, every imperfect recall agent, regardless of its abstraction features, outperformed every perfect recall agent. In terms of exploitability, aside from IR-PHS-PHS, every CFR-BR agent using imperfect recall was found to be less exploitable than its perfect recall equivalent. While CFR-BR is not theoretically guaranteed to converge to a least exploitable strategy in an imperfect recall game, our results provide an upper bound: the least exploitable IR-KE-KO strategy is exploitable for at most 64.820 mbb/g, far less than the least exploitable perfect recall agent. While Waugh *et al.* found that imperfect recall and additional features provided a small advantage, we have shown a significant improvement while using the same domain features.

Finally, the CFR and CFR-BR results presented in Table 5.4 support Johanson

et al. 's proposed use of CFR-BR to evaluate strategies instead of measuring the exploitability of abstract game Nash equilibria. The CFR results are inconsistent, showing no clear advantage for perfect or imperfect recall, and ordering the agents differently than the one-on-one crosstable. While there is no guarantee that the one-on-one results and exploitability should agree, the CFR-BR strategies are both far less exploitable in all cases, show an advantage for imperfect recall, and rank the top three agents in the same order as the one-on-one results. Using CFR-BR to evaluate abstractions based on their ability to approximate an unabstracted Nash equilibrium provides a more consistent metric than the previous approaches.

5.7 Conclusion

Historically, state-space abstraction techniques in extensive-form games have been evaluated by computing optimal abstract strategies and comparing their one-on-one performance and exploitability. A recently published technique, CFR-BR, directly finds the abstract strategy with the lowest real game exploitability, providing a more consistent measure of an abstraction's quality. Using this technique, we evaluated two abstraction choices in the poker domain: expectation-based as opposed to distribution-aware distance metrics, and imperfect recall abstractions. Our findings on distribution-aware techniques support those of Gilpin and Sandholm: distribution-aware distance metrics provide a clear advantage once the abstract game is large enough. We also demonstrated a clear improvement in one-on-one performance and exploitability through the use of imperfect recall abstractions, and demonstrated that imperfect recall abstractions can contain less exploitable strategies than equal sized perfect recall strategies.

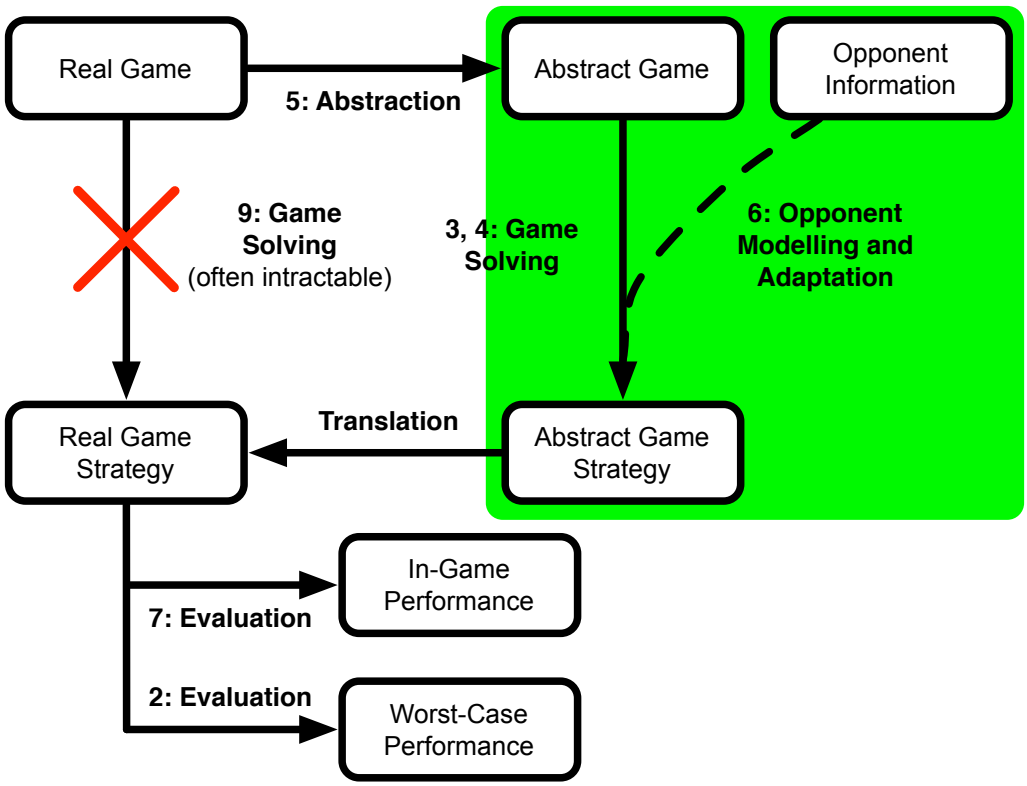
Acknowledgments

We would like to thank Mihai Ciucu, Eric Jackson, Mengliao Wang, and the members of the University of Alberta Computer Poker Research Group. This research was supported by NSERC and Alberta Innovates Technology Futures, and was made possible by the computing resources provided by WestGrid, Réseau Québécois de Calcul de Haute Performance, and Compute/Calcul Canada.

Chapter 6

Opponent Modelling and Counter Strategies

Data Biased Robust Counter Strategies
Michael Johanson and Michael Bowling
Presented at AISTATS 2009 [51]



Thus far, we have described techniques for approximating Nash equilibria. In a two-player zero-sum game, a Nash equilibrium is a useful default strategy as it is guaranteed to not lose on expectation to any opponent, even a worst-case omniscient opponent who knows our agent's strategy before the match begins, or who can quickly model it and respond to it online. However, the opponents that our agents will face in practice will be humans and computer agents whose strategies will be imperfect and exploitable. Instead of cautiously trying to not lose, our agents can instead better achieve the goal of the game by modelling their opponent and developing an exploitive counter-strategy, and hopefully increase their expected winnings far beyond what a Nash equilibrium would win.

However, opponent modelling and adaptation is a difficult and error-prone task. If we attempt to model our opponent offline through observations of their past behaviour, then we must generalize a relatively small number of observations across a vastly larger strategy space. Further, the natural inclination to compute and use a utility-maximizing best response strategy is dangerous. Best response strategies tend to be **brittle**: they maximize utility against a very particular opponent, but if our model is imperfect due to a limited number of observations, or if the opponent's strategy is changing over time, then it can itself be badly exploited.

In prior work, we proposed using **robust** counter strategies, that provide a tradeoff between minimizing exploitability (like a Nash equilibrium) and maximizing exploitation (like a best response). The algorithm we used, Restricted Nash Response (RNR) [54], generates the pareto-optimal set of strategies that trade off between these goals, in practice outperforming any simple mixture between a Nash equilibrium and a best response. However, RNR requires complete knowledge of the opponent's strategy, and is error prone when using a model built from observations.

In this chapter's paper, we present the Data Biased Response algorithm. Data Biased Response is a variant of CFR that restricts one player's behaviour at individual information sets, based on a provided model that we have constructed. The degree to which the model is enforced is dependent on our confidence in the model's accuracy at each decision point, to avoid errors caused by sparse data. The result is a practical, efficient offline algorithm for computing robust counter-strategies.

DBR forms part of our current approach for opponent modelling and online adaptation, called the Implicit Modelling framework. Developed by Bard, Bowling, myself and our colleagues [8; 7; 9], this framework anticipates types of opponents we might face, clusters them into categories, uses DBR to construct counter-strategies to each category, and then uses the Imaginary Observations technique we will discuss in Chapter 7 to switch between these counter-strategies online.

Author's contributions. I developed the original idea for this paper, which addresses many of the faults of our earlier work on Restricted Nash Response [54]. I designed and implemented the algorithm and performed all of the experiments. The results were interpreted by Bowling and myself. Bowling contributed the formalism in Section 6.7 which casts DBR as a task of picking a robust prior. The paper was written and edited equally by Bowling and myself.

Data Biased Robust Counter Strategies[1][2]

Michael Johanson (johanson@cs.ualberta.ca)
Michael Bowling (bowling@cs.ualberta.ca)

Abstract:
The problem of exploiting information about the environment while still being robust to inaccurate or incomplete information arises in many domains. Competitive imperfect information games where the goal is to maximally exploit an unknown opponent's weaknesses are an example of this problem. Agents for these games must balance two objectives. First, they should aim to exploit data from past interactions with the opponent, seeking a best-response counter strategy. Second, they should aim to minimize losses since the limited data may be misleading or the opponent's strategy may have changed, suggesting an opponent-agnostic Nash equilibrium strategy. In this paper, we show how to partially satisfy both of these objectives at the same time, producing strategies with favourable tradeoffs between the ability to exploit an opponent and the capacity to be exploited. Like a recently published technique, our approach involves solving a modified game; however the result is more generally applicable and even performs well in situations with very limited data. We evaluate our technique in the game of two-player, Limit Texas Hold'em.

6.1 Introduction

Maximizing utility in the presence of other agents is a fundamental problem in game theory. In a zero-sum game, utility comes from the exploitation of opponent weaknesses, but it is important not to allow one's own strategy to be exploited in turn. Two approaches to such problems are well known: best response strategies and Nash equilibrium strategies. A best response strategy maximizes utility for an agent, assuming perfect knowledge of its static opponent. However, such strategies are *brittle*: against a worst case opponent, they have a high exploitability. In a two-player zero-sum game, a Nash equilibrium strategy maximizes its utility against a worst-case opponent. As a result, we say that such strategies are *robust*. If a perfect model of the opponent is available, then they can be exploited by a best response; if a model is not available, then playing a Nash equilibrium strategy is a sensible choice. However, if a model exists but it is somewhat unreliable (*e.g.*, if it is formed from a limited number of observations of the opponent's actions, or if the opponent is known to be changing strategies) then a better option may be to compromise: accepting a slightly lower worst-case utility in return for a higher utility if the model is approximately correct.

[1]The paper presented in this chapter originally appeared at the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09). Copyright 2010 by the authors. M. Johanson and M. Bowling. Data Biased Robust Counter Strategies. Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, 264-271, 2009.

[2]The appendix has additional results related to this paper.

One simple approach for creating such a compromise strategy is to create both a best response strategy and a Nash equilibrium strategy, and then play a mixture of the two. Before each game, we will flip a biased coin. With probability p we will use the best response, and with probability $(1-p)$ we will use the Nash equilibrium. By varying p, we can create a range of strategies that linearly trade off *exploitation* of the opponent and our own *exploitability* by a worst-case opponent. While this approach is a useful baseline, we would like to make more favourable tradeoffs between these goals.

McCracken and Bowling proposed ϵ-safe strategies as another approach [66]. The set of ϵ-safe strategies contains all strategies that are exploitable by no more than ϵ. From this set, the strategies that maximize utility against the opponent are the set of ϵ-safe best responses. Thus, for a chosen ϵ, the set of ϵ-safe best responses achieve the best possible tradeoffs between exploitation and exploitability. However, their approach is computationally infeasible for large domains, and has only been applied to Ro-Sham-Bo (Rock-Paper-Scissors).

In previous work we proposed the restricted Nash response [54] technique (RNR) as a practical approach for generating a range of strategies that provide good tradeoffs between exploitation and exploitability. In this approach, a modified game is formed in which the opponent is forced to act according to an opponent model with some probability p, and is free to play the game as normal with probability $(1 - p)$. When p is 0 the result is a Nash equilibrium, and when p is 1 the result is a best response. When $0 < p < 1$ the technique produces a counter-strategy that provides different tradeoffs between exploitation and exploitability. In fact, the counter-strategies generated are in the set of ϵ-safe best responses for the counter-strategy's value of ϵ, making them the best possible counter-strategies, assuming the model is correct. In a practical setting, however, the model is likely formed through a limited number of observations of the opponent's actions, and it may be incomplete (it cannot predict the opponent's strategy in some states) or inaccurate. As we will show in this paper, the restricted Nash response technique can perform poorly under such circumstances.

In this paper, we present a new technique for generating a range of counter-strategies that form a compromise between the exploitation of a model and its exploitability. These counter-strategies, called **data biased responses** (DBR), are more resilient to incomplete or inaccurate models than the restricted Nash response (RNR) counter-strategies. DBR is similar to RNR in that the technique involves computing a Nash equilibrium strategy in a modified game where the opponent is forced with some probability to play according to a model. Unlike RNR, the opponent's strategy is constrained on a per-information set basis, and depends on our confidence in the accuracy of the model. For comparison to the RNR technique, we demonstrate the effectiveness of the technique in the challenging domain of 2-player Limit Texas Hold'em Poker.

6.2 Background

A **perfect information extensive game** consists of a tree of game states and terminal nodes. At each game state, an action is taken by one player (or by "chance") causing a transition to a child state; this is repeated until a terminal state is reached. The terminal state defines the payoffs to the players. In **imperfect information extensive games** such as poker, the players cannot observe some piece of information (such as their opponent's cards) and so they cannot exactly determine which game state they are in. Each set of indistinguishable game states is called an **information set** and we denote such a set by $I \in \mathcal{I}$. A **strategy** for player i, σ_i, is a mapping from information sets to a probability distribution over actions, so $\sigma_i(I, a)$ is the probability player i takes action a in information set I. The space of all possible strategies for player i will be denoted Σ_i. In this paper, we will focus on two player games.

Given strategies for both players, we define $u_i(\sigma_1, \sigma_2)$ to be the expected utility for player i if player 1 uses the strategy $\sigma_1 \in \Sigma_1$ and player 2 uses the strategy $\sigma_2 \in \Sigma_2$. A best response to an opponent's strategy σ_2 is a strategy for player 1 that achieves the maximum expected utility of all strategies when used against the opponent's strategy. There can be many strategies that achieve the same expected utility; we refer to the set of best responses as $BR(\sigma_2) \subseteq \Sigma_1$. For example, the set of best responses for player 1 to use against σ_2 is defined as:

$$BR(\sigma_2) = \{\ \sigma_1 \in \Sigma_1 : \\ \forall \sigma_1' \in \Sigma_1 u_1(\sigma_1, \sigma_2) \geq u_1(\sigma_1', \sigma_2)\}$$

A **strategy profile** σ consists of a strategy for each player in the game; *i.e.*, (σ_1, σ_2). In the special case where $\sigma_1 \in BR(\sigma_2)$ and $\sigma_2 \in BR(\sigma_1)$, we refer to σ as a Nash equilibrium. A **zero-sum extensive game** is an extensive game where $u_1 = -u_2$ (one player's gains are equal to the other player's losses). In such games, all Nash equilibrium strategies have the same utility for the players, and we refer to this as the **value of the game**. We define the term **exploitability** to refer to the difference between a strategy's utility when playing against its best-response and the value of the game for that player. We define **exploitation** to refer to the difference in utility between one strategy's utility against a specific opponent strategy and the value of the game for that player.

A strategy that can be exploited for no more than ϵ is called **ϵ-safe**, and is a member of the set of ϵ-safe strategies $\Sigma_1^{\epsilon\text{-safe}} \subseteq \Sigma_1$. A strategy profile where each strategy can be exploited by no more than ϵ is called an ϵ-Nash equilibrium. Given the set $\Sigma_1^{\epsilon\text{-safe}}$, there is a subset $BR^{\epsilon\text{-safe}}(\sigma_2) \subseteq \Sigma_1^{\epsilon\text{-safe}}$ that contains the strategies that maximize utility against σ_2:

$$BR^{\epsilon\text{-safe}}(\sigma_2) = \{\ \sigma_1 \in \Sigma^{\epsilon\text{-safe}} : \\ \forall \sigma_1' \in \Sigma^{\epsilon\text{-safe}} u_1(\sigma_1, \sigma_2) \geq u_1(\sigma_1', \sigma_2)\}$$

6.3 Texas Hold'em Poker

Heads-Up Limit Texas Hold'em poker is a two-player wagering card game. In addition to being commonly played in casinos (both online and in real life), it is also the main event of the AAAI Computer Poker Competition [65], an initiative to foster research into AI for imperfect information games. Texas Hold'em is a very large zero-sum extensive form game with imperfect information (the opponent's cards are hidden) and stochastic elements (cards are dealt at random). Each individual game is short, and players typically play a session of many games.

We will briefly summarize the rules of the game. A session starts with each player having some number of **chips**, which usually represent money. A single game of Heads-Up Limit Texas Hold'em consists of each player being forced to place a small number of chips (called a **blind**) into the **pot** before being dealt two private cards. The players will combine these private cards with five public cards that are revealed as the game progresses. The game has four phases: the preflop (when two private cards are dealt), the flop (when three public cards are dealt), the turn (when one public card is dealt) and the river (when one final public card is dealt). If both players reach the end of the game (called a **showdown**), then both players reveal their private cards and the player with the best 5-card poker hand wins all of the chips in the pot. If only one player remains in the game, then that player wins the pot without revealing their cards. After the cards are dealt in each phase, the players engage in a **round of betting**, where they **bet** by placing additional chips in the pot that their opponent must match or exceed in order to remain in the game. To do this, the players alternate turns and take one of three actions. They may **fold** to exit the game and let the opponent win, **call** to match the opponent's chips in the pot, or **raise** to match, and then add a fixed number of additional chips (the "bet" amount). When both players have called, the round of betting is over, and no more than four bets are allowed in a single round.

The goal is to win as much money as possible from the opponent by the end of the session. This distinguishes poker from games such as Chess or Checkers where the goal is simply to win and the magnitude of the win is not measured. The performance of an agent is measured by the number of bet amounts (or just bets) they win per game across a session. Between strong computer agents, this number can be small, so we present the performance in millibets per game (mbb/g), where a millibet is one thousandth of a bet[3]. A player that always folds will lose 750 millibets per game to their opponent, and a strong player can hope to win 50 millibets per game from their opponent. Due to a standard deviation of approximately 6000 millibets per game, it can take more than one million games to distinguish with 95% confidence a difference of 10 millibets per game.

Since the goal of the game is to maximize the exploitation of one's opponent,

[3]From 2006 to 2010, "millibet" or "millibets per game" was the unit commonly used by researchers. The switch to "milliblinds per game" began in 2011 with the Accelerated Best Response paper [53]. In heads-up limit Texas hold'em, millibets and milliblinds are equivalent. The milliblinds unit is preferred as it extends to games such as no-limit Texas hold'em, where there is no such "big bet" size defined by the rules, whereas all poker games have blinds or antes.

the game emphasizes the role of exploitive strategies as opposed to equilibrium strategies. In the two most recent years of the AAAI Computer Poker Competition, the "Bankroll" event which rewards exploitive play has been won by agents that lost to some opponents, but won enough money from the weakest agents to have the highest total winnings. However, many of the top agents have been designed to take advantage of a suspected *a priori* weakness common to many opponents. A more promising approach is to observe an opponent playing for some fixed number of games, and use these observations to create a counter-strategy that exploits the opponent for more money than a baseline Nash equilibrium strategy or a strategy that exploits some expected weaknesses.

6.3.1 Abstraction

The variant of poker described above has 9.17×10^{17} game states; computing best responses and Nash equilibria in a game of this size is intractable. Therefore, it is common practise to instead reduce the real game to a much smaller abstract game that maintains as many of the strategic properties as possible. The strategies of interest to us will be computed in this abstract game. To use the abstract game strategy to play the real game, we will map the current real game information set to an abstract game information set, and choose the action specified by the abstract game strategy.

The game is abstracted by merging information sets that result from similar chance outcomes. On the preflop, one such abstraction might reduce the number of chance outcomes from 52 choose 2 down to 5, and from (52 choose 2)(50 choose 3) to 25 on the flop. Each chance outcome is reduced to one of 5 outcomes, giving 625 possible combinations, resulting in a game that has 6.45×10^9 game states. In this abstract game, best response counter-strategies can be computed in time linear in the size of the game tree; on modern hardware, this takes roughly 10 minutes. Using recent advances for solving extensive form games [110], a Nash equilibrium for this abstract game can be approximated to within 3 millibets per game in under 10 hours.

6.3.2 Opponent Strategies

Much of the recent effort towards creating strong agents for Texas Hold'em has focused on finding Nash equilibrium strategies for abstract games [110; 34]. We want to examine the ability to exploit opponent weaknesses, so we will examine results where the opponent is not playing an equilibrium strategy. Toward this end, we created an agent similar to "Orange", which was designed to be overly aggressive but still near equilibrium and competed in the First Man-Machine Poker Championship [47, p. 82],. "Orange" is a strategy for an abstract non-zero-sum poker game where the winner gets 7% more than usual, while the loser pays the normal price. When this strategy is used to play the normal (still abstract) zero-sum game of poker, it is exploitable for 28 millibets per game. This value is the upper bound

on the performance obtainable by any counter-strategy that plays in the same abstraction.

In this paper, we will also refer to an agent called "Probe" [54]. Probe is a trivial agent that never folds, and calls and raises whenever legal with equal probability. The Probe agent is useful for collecting observations about an opponent's strategy, since it forces them into all of the parts of the game tree that the opponent will consent to reach.

6.3.3 Opponent Beliefs

A belief about the opponent's current strategy can simply be encoded as a strategy itself. Even a posterior belief derived from a complicated prior and many observations still can be summarized as a single function mapping an information set to a distribution over actions, the **expected posterior strategy**[4]. In this work, we will mainly take a frequentist approach to observations of the opponent's actions (although we discuss a Bayesian interpretation to our approach in Section 6.7). Each observation is one full information game of poker: both players' cards are revealed. The model of our opponent will consider all of the information sets in which we have observed the opponent acting. The probability of the opponent model taking an action a in such an information set I is then set to the ratio of the number of observations of the opponent playing a in I to the number of observations of I. There will likely be information sets in which we have never observed the opponent acting. For such information sets, we establish a **default policy** to always choose the call action [47, p. 60][5]

Since our opponent model is itself a strategy, it can be used to play against the counter-strategies that are designed to exploit it. We would expect the counter-strategies to perform very well in such cases, and this is demonstrated in our previous work on restricted Nash responses [54]. However, since the model is constructed only from (possibly a small number) observations of the opponent's strategy, it is more interesting to examine how the counter-strategies perform against the actual opponent's strategy.

6.4 Limitations of Current Methods

As discussed in the introduction, restricted Nash response counter-strategies form an envelope of possible counter-strategies to use against the opponent, assuming the opponent model is correct [54]. The restricted Nash response technique was designed to solve the brittleness of best response strategies. As was presented in Table 1 of that work, best response strategies perform well against their intended opponent, but they can perform very badly against other opponents, and are highly

[4] If $f : \Sigma_2 \rightarrow \Re$ is the posterior density function over strategies, then the expected posterior strategy chooses action a at information set I with probability $\bar{\sigma}_1(I, a) = \int_{\sigma_1 \in \Sigma_1} \sigma_1(I, a) f(\sigma_1)$

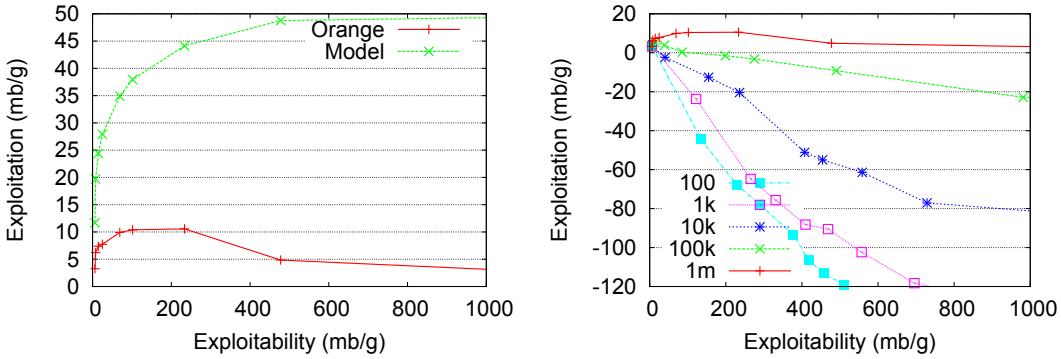
[5] Alternative default policies were tried in this previous work, but all performed far inferior.

exploitable by a worst-case opponent. Restricted Nash response strategies are robust, and any new technique for producing counter-strategies should also be able to produce robust strategies. However, restricted Nash response strategies have three limitations. We will show that our new counter-strategy technique addresses these issues.

Before discussing the limitations, we first explain the exploitability-versus-exploitation graph that is used throughout the paper. For each counter-strategy, we can measure the exploitability (worst-case performance) and exploitation (performance against a specific opponent). So we can plot any counter-strategy as a point on a graph with these axes. Restricted Nash responses involve a family of counter-strategies attained by varying p. Hence, we plot a curve passing through a set of representative p-values to demonstrate the shape of the envelope of strategies. Since the exploitability is determined by the choice of p, we are (indirectly) controlling the exploitability of the resulting counter-strategy, and so it appears on the x-axis; the counter-strategy's exploitation of the specific opponent is the result, and is shown on the y-axis. In each of the following graphs, the values of p used were 0, 0.5, 0.7, 0.8, 0.9, 0.93, 0.97, 0.99, and 1. Each value of p corresponds to one datapoint on each curve. Unless otherwise stated, each set of counter-strategies was produced with 1 million observed games of Orange playing against Probe.

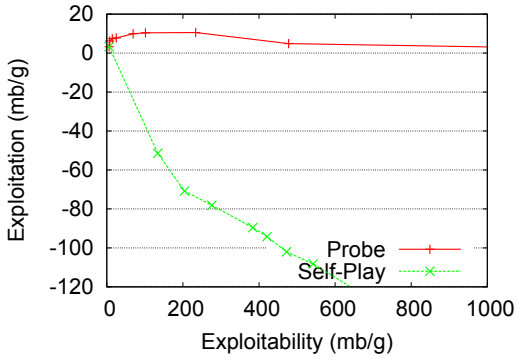
Restricted Nash response counter-strategies can overfit to the model. By varying p, the resulting restricted Nash response counter-strategies each present a different tradeoff of exploitation and exploitability when compared against their opponent model. As p increases, the counter-strategies exploit the opponent model to a higher degree, and are themselves more exploitable. However, as Figure 6.1a shows, this trend does not hold when we compare their performance against the actual opponent instead of the opponent model. As p increases, the counter-strategies begin to do markedly worse against the actual Orange strategy. The computed counter-strategy has overfit to the opponent model. As the number of observations approach the limit, the opponent model will perfectly match the actual opponent in the reachable part of the game tree, and this effect will lessen. In a practical setting, however, p must be chosen with care so that the resulting counter-strategies provide favourable trade-offs.

Restricted Nash response counter-strategies require a large quantity of observations. It is intuitive that, as any technique is given more observations of an opponent, the counter-strategies produced will grow in strength. This is true of the restricted Nash response technique. However, if there is not a sufficient quantity of observations, increasing p can make the resulting counter-strategies *worse* than the equilibrium strategy. This is another aspect of the restricted Nash response technique's capacity to overfit the model; if there is an insufficient number of observations, then the default policy plays a larger part of the model's strategy and the resulting counter-strategy is less applicable to the actual opponent. Figure 6.1b shows this effect. With less than 100 thousand observed games, increasing p causes



(a) Overfitting

(b) Observations



(c) Training

Figure 6.1: Exploitation versus exploitability curves that illustrate three problems in the restricted Nash response technique. In 6.1a, we note the difference in performance when counter-strategies play against the opponent model and against the actual opponent. In 6.1b, we see how a scarcity of observations results in poor counter-strategies. In 6.1c, we see that the technique performs poorly when self-play data is used. Note that the red, solid curve is the same in each graph.

the counter-strategies to be both more exploitable and less exploitive.

Restricted Nash response counter-strategies are sensitive to the choice of training opponent. Ideally, a technique for creating counter-strategies based on observations should be able to accept any reasonably diverse set of observations as input. However, the restricted Nash response technique requires a very particular set of observations in order to perform well. Figure 6.1c shows the performance of two sets of restricted Nash response counter-strategies. The set labelled Probe uses an opponent model that observed one million games of Orange playing against Probe; the set labelled Self-Play uses an opponent model that observed one million games of Orange playing against itself. One might think that a model constructed from self-play observations would be ideal, because it would be accurate in the parts of the game tree that the opponent is likely to reach. Instead, we find that self-play data is of no use when constructing a restricted Nash response counter-strategy. If an agent will not play to reach some part of the game tree, then the opponent model has no observations of the opponent in that part of the tree, and is forced to turn to the default policy which may be very dissimilar from the actual opponent's strategy. The Probe agent forces the opponent to play into all of the parts of the tree reachable because of the opponent's strategy, however, and thus the default policy is used less often.

6.5 Data Biased Response

The guiding idea behind the restricted Nash response technique is that the opponent model may not be perfect. The parameter p can be thought of as a measure of confidence in the model's accuracy. Since the opponent model is based on observations of the opponent's actions, there can be two types of flaws in the opponent model. First, there may be information sets in which we never observed the opponent, and so the opponent model must provide a default policy to be taken at this information set. Second, in information sets for which there were a small number of observations, the observed frequency of actions may not match the true opponent's action probabilities.

We claim that the restricted Nash response technique's selection of one parameter, p, is not an accurate representation of the problem, because the accuracy of the opponent model is not uniform across all of the reachable information sets. Consider the two cases described above. First, in unobserved information sets, the opponent model uses the default policy and is unlikely to accurately reflect the opponent's strategy. If we could select a value of p for just this information set, then p would be 0. Second, the number of observations of a particular information set will vary wildly across the game tree. In information sets close to the root, we are likely to have many observations, and so we expect the model to be accurate. In information sets that are far from the root, we will tend to have fewer observations, and so we expect the model to be less accurate. If we were selecting a value of p for one information set, it should depend on how accurate we expect the model to be;

one measure of this is the number of times we have observed the opponent acting in this information set.

This is the essential difference between the restricted Nash response technique and the data biased response technique. Instead of choosing one probability p that reflects the accuracy of the entire opponent model, we will assign one probability to each information set I and call this mapping P_{conf}. We will then create a modified game in the following way. Whenever the restricted player reaches I, they will be forced to play according to the model with probability $P_{\text{conf}}(I)$, and can choose their actions freely with probability $(1-P_{\text{conf}}(I))$. The other player has no restrictions on their actions. When we solve this modified game, the unrestricted player's strategy becomes a robust counter-strategy to the model.

One setting for P_{conf} is noteworthy. If $P_{\text{conf}}(I)$ is set to 0 for some information sets, then the opponent model is not used at all and the player is free to use any strategy. However, since we are solving the game, this means that we assume a worst-case opponent and essentially compute a Nash equilibrium in these subgames.

6.5.1 Solving the Game

Given an opponent model σ_{fix} and P_{conf}, the restricted player chooses a strategy σ_2' that makes up part of their restricted strategy σ_2. The resulting probability of σ_2 taking action a at information set I is given as:

$$\sigma_2(I,a) = P_{\text{conf}}(I) \times \sigma_{\text{fix}}(I,a) + (1 - P_{\text{conf}}(I)) \times \sigma_2'(I,a) \tag{6.1}$$

Define $\Sigma_2^{P_{\text{conf}},\sigma_{\text{fix}}}$ to be the set of strategies for the restricted player, given the possible settings of σ_2'. Among this set of strategies, we can define the subset of best responses to an opponent strategy σ_1, $BR^{P_{\text{conf}},\sigma_{\text{fix}}}(\sigma_1) \subseteq \Sigma_2^{P_{\text{conf}},\sigma_{\text{fix}}}$. Solving a game with the opponent restricted accordingly, finds a strategy profile (σ_1^*, σ_2^*) that is a restricted equilibrium, where $\sigma_1^* \in BR(\sigma_2^*)$ and $\sigma_2^* \in BR^{P_{\text{conf}},\sigma_{\text{fix}}}(\sigma_1)$. In this pair, the strategy σ_1^* is a **$\boldsymbol{P_{\text{conf}}}$-restricted Nash response to the opponent model $\boldsymbol{\sigma_{\text{fix}}}$**, which we call a data biased response counter-strategy.

6.5.2 Choosing P_{conf}

We will now present four ways in which P_{conf} can be chosen, all of which have two aspects in common. First, each approach sets $P_{\text{conf}}(I)$ for an information set I as a function of the number of observations we have of the opponent acting in information set I, n_I. As the number of observations of our opponent acting in I increase, we will become more confident in the model's accuracy. If $n_I = 0$, then we set $P_{\text{conf}}(I)$ to zero, indicating that we have no confidence in the model's prediction. Note that this choice in setting P_{conf} removes the need for a default policy. As mentioned in Section 6.5, this means the restricted player will become a worst-case opponent in any information sets for which we have no observations. Second, each approach accepts an additional parameter $P_{\max} \in [0,1]$, which acts

in a similar fashion to p in the restricted Nash response technique. It is used to set a maximum confidence for P_{conf}. Varying $P_{\max}$ in the range $[0, 1]$ allows us to set a tradeoff between exploitation and exploitability, while n_I indicates places where our opponent model should not be trusted.

Removing the default strategy. First, we consider a simple choice of P_{conf}, which we call the 1-Step function. In information sets where we have never observed the opponent, P_{conf} returns 0; otherwise, it returns $P_{\max}$. This choice of P_{conf} allows us to isolate the modelling error caused by the default policy from the error caused by the opponent model's action probabilities not matching the action probabilities of the actual opponent.

Requiring more observations. Second, we consider another simple choice of P_{conf}, which we call the 10-Step function. In information sets where we have observed the opponent fewer than 10 times, P_{conf} returns 0; otherwise, it returns $P_{\max}$. Thus, it is simply a step function that requires ten observations before expressing any confidence in the model's accuracy.

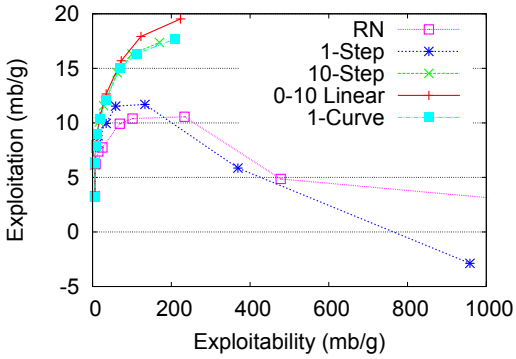
Linear confidence functions. Third, we consider a middle ground between our two step functions. The 0-10 Linear function returns $P_{\max}$ if $n_I > 10$ and $(n_I \times P_{\max})/10$ otherwise. Thus, as we obtain more observations, the function expresses more confidence in the accuracy of the opponent model.

Curve confidence functions. Fourth, we consider a setting of P_{conf} with a Bayesian interpretation. The s-Curve function returns $P_{\max} \times (n_I/(s + n_I))$ for any constant s; in this experiment, we used $s = 1$. Thus, as we obtain more observations, the function approaches $P_{\max}$. The foundation for this choice of P_{conf} is explained further in Section 6.7.

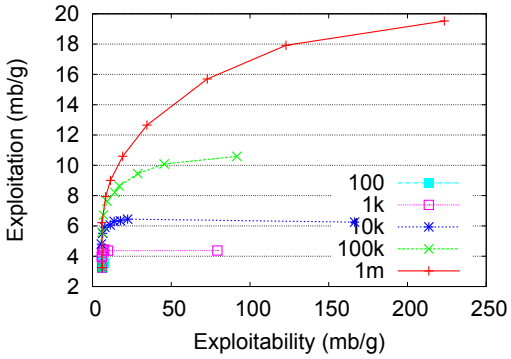
6.6 Results

In Section 6.3, we presented three problems with restricted Nash response strategies. In this section, we will revisit these three problems and show that data biased response counter-strategies overcome these weaknesses. In each experiment, the sets of restricted Nash response and data biased response counter-strategies were created with p and $P_{\max}$ (respectively) parameters of 0, 0.5, 0.7, 0.8, 0.9, 0.93, 0.97, 0.99, and 1. Unless otherwise stated, each set of counter-strategies was produced with 1 million observed games of Orange playing against Probe.

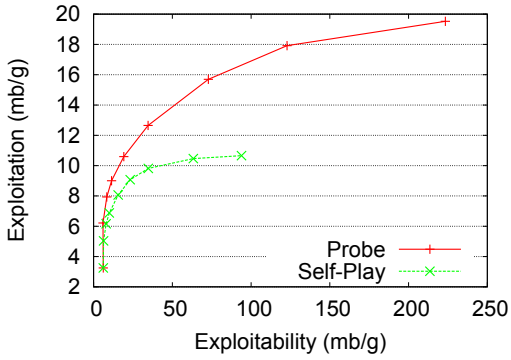
Overfitting to the model. We begin with the problem of overfitting to the model. Figure 6.2a shows the results of sets of restricted Nash response and 1-Step, 10-Step and 0-1 Linear data biased response counter-strategies playing against Orange



(a) Overfitting



(b) Observations



(c) Training

Figure 6.2: Exploitation versus exploitability curves for data biased response counter-strategies. 6.2a shows that restricted Nash and 1-Step counter-strategies overfit the model, while 10-Step, 0-10 Linear, and 1-Curve counter-strategies do not. 6.2b shows that the 0-10 Linear counter-strategies are effective with any quantity of training data. 6.2c shows that the 0-10 Linear counter-strategies can accept any type of training data. Note that the red, solid curve is the same in each graph.

and the opponent model of Orange. Two of the results are noteworthy. First, we observe that the set of 1-Step data biased response counter-strategies overfit the model. Since the 1-Step data biased response counter-strategies did not use the default policy, this shows us that the error caused by the opponent model's action probabilities not agreeing with the actual opponent's action probabilities is a nontrivial problem and that the default policy is not the only weakness. Second, we notice that the 0-10 Linear, 10-Step and 1-Curve data biased response counter-strategies do not overfit the opponent model, even at the last datapoint where $P_{\max}$ is set to 1.

Quantity of observations. Next, we examine the problem of the quantity of observations necessary to produce useful counter-strategies. In Figure 6.1b, we showed that with insufficient quantities of observations, restricted Nash counter-strategies not only did not exploit the opponent but in fact performed worse than a Nash equilibrium strategy (which makes no attempt to exploit the opponent). In Figure 6.2b, we show that the 0-10 Linear data biased response counter-strategies perform well, regardless of the quantity of observations provided. While the improvement in exploitation from having 100 or 1000 observations is very small, for $P_{\max} < 1$ the counter-strategies became only marginally more exploitable. This is a marked difference from the restricted Nash response results in Figure 6.1b.

Source of observations. Finally, we consider the problem of the source of the observations used to create the model. In Figure 6.1c, we showed that the restricted Nash response technique required observations of the opponent playing against an opponent such as Probe in order to create useful counter-strategies. In Figure 6.2c, we show that while the data biased response counter-strategies produced are more effective when the opponent model observes games against Probe, the technique does still produce useful counter-strategies when provided with self-play data.

6.7 Discussion

We motivated data biased responses by noting that the confidence in our model is not uniform over all information sets, and suggesting p should be some increasing function of the number of observations at a particular information set. We can give an alternative motivation for this approach by considering the framework of Bayesian decision making. In the Bayesian framework we choose a prior density function ($f : \Sigma_2 \rightarrow \Re$) over the unknown opponent's strategy. Given observations of the opponent's decisions $\mathcal{Z}$ we can talk about the posterior probability $\Pr(\sigma_2|\mathcal{Z}, f)$. If only one more hand is to be played, decision theory instructs us to maximize our expected utility given our beliefs.

$$\operatorname*{argmax}_{\sigma_1} \int_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) \Pr(\sigma_2|\mathcal{Z}, f) \tag{6.2}$$

Since utility is linear in the sequence form representation of strategy, we can move the integral inside the utility function allowing us to solve the optimization as the best-response to the expected posterior strategy (see Footnote 4).

However, instead of choosing a single prior density, suppose we choose a set of priors (F), and we want to play a strategy that would have large utility for anything in this set. A traditional Bayesian approach might require us to specify our uncertainty over priors from this set, and then maximize expected utility given such a hierarchical prior. Suppose, though, that we have no basis for specifying such a distribution over distributions. An alternative then is to maximize utility in the worst case.

$$\operatorname*{argmax}_{\sigma_1} \min_{f \in F} \int_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) \Pr(\sigma_2 | \mathcal{Z}, f) \tag{6.3}$$

In other words, employ a strategy that is robust to the choice of prior. Notice that if F contains a singleton prior, this optimization is equivalent to the original decision theoretic approach, *i.e.* , a best response strategy. If F contains all possible prior distributions, then the optimization is identical to the game theoretic approach, *i.e.* , a Nash equilibrium strategy. Other choices of the set F admit optimizations that trade-off exploiting data with avoiding exploitation.

Theorem 8 *Consider F to be the set of priors composed of independent Dirichlet distributions for each information set, where the strength (sum of the Dirichlet parameters) is at most s. The strategy computed by data biased response when $P_{\text{conf}}(I) = n_I/(s + n_I)$ is the solution to the optimization in 6.3.*

Proof *(Sketch)* If we let Σ_2^s be the set of resulting expected posterior strategies for all choices of priors $f \in F$. It suffices to show that $\Sigma_2^s = \Sigma^{P_{\text{conf}}, \sigma_{\text{fix}}}$. For any prior $f \in F$, let $\alpha_{I,a}^f$ be the Dirichlet weight for the outcome a at information set I. Let $\sigma_{\text{fix}}(I, a) = \alpha_{I,a}^f / \sum_{a'} \alpha_{I,a'}^f$, in other words the strategy where the opponent plays the expected prior strategy when given the opportunity. The resulting expected posterior strategy is the the same as σ_2 from Equation 6.1 and so is in the set $\Sigma^{P_{\text{conf}}, \sigma_{\text{fix}}}$. Similarly, given σ_{fix} associated with a strategy σ_2 in $\Sigma^{P_{\text{conf}}, \sigma_{\text{fix}}}$, let $\alpha_{I,a} = s\sigma_{\text{fix}}(I, a)$. The resulting expected posterior strategy is the same as σ_2. The available strategies to player 2 are equivalent, and so the resulting min-max optimizations are equivalent. □

In summary, we can choose P_{conf} in data biased response so that it is equivalent to finding strategies that are robust to a set of independent Dirichlet priors.

6.8 Conclusion

The problem of exploiting information about a suspected tendency in an environment while minimizing worst-case performance occurs in several domains, and becomes more difficult when the information may be limited or inaccurate. We

reviewed restricted Nash response counter-strategies, a recent work on the opponent modelling interpretation of this problem in the Poker domain, and highlighted three shortcomings in that approach. We proposed a new technique, data biased responses, for generating robust counter-strategies that provide good compromises between exploiting a tendency and limiting the worst case exploitability of the resulting counter-strategy. We demonstrated that the new technique avoids the three shortcomings of existing approaches, while providing better performance in the most favourable conditions for the existing approaches.

6.9 Acknowledgments

We would like to thank the members of the University of Alberta Computer Poker Research Group. This research was supported by NSERC and iCore.

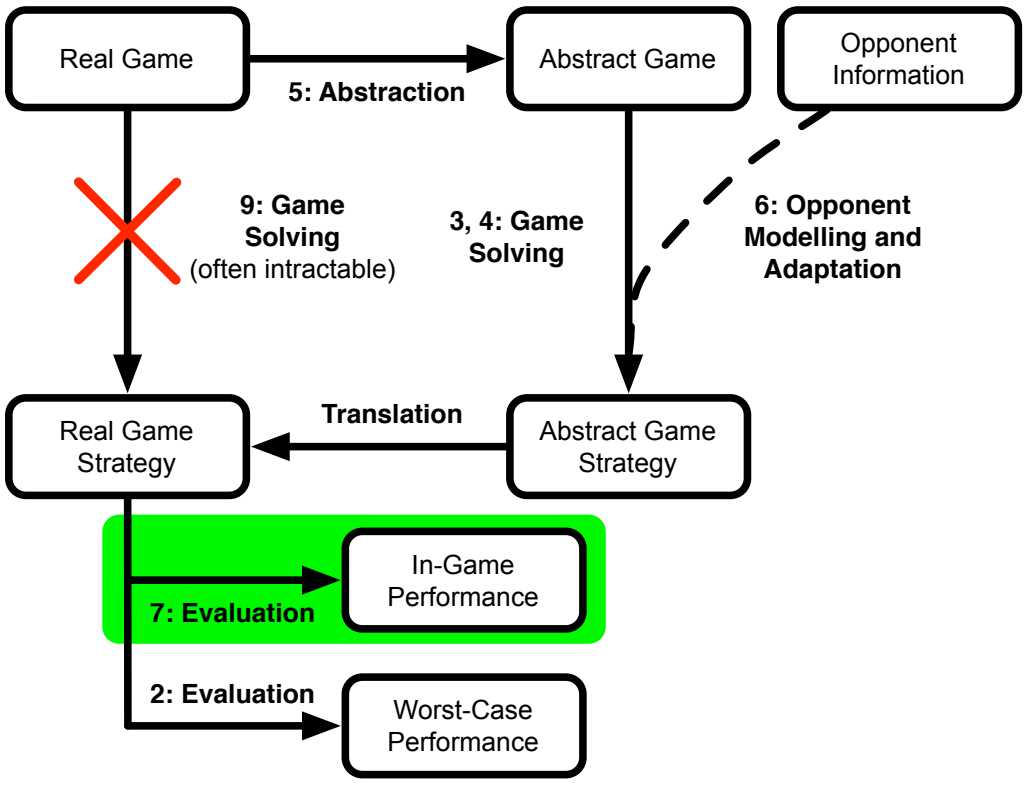
Chapter 7

In-Game Evaluation

Strategy Evaluation in Extensive Games with Importance Sampling

Michael Bowling, Michael Johanson, Neil Burch, Duane Szafron

Presented at ICML 2008 [19]



Estimating an agent's peformance through observations is difficult because of random chance and hidden information. In a game such as poker, these can be caused by random aspects of the game (*i.e.* cards being dealt randomly), agents sampling actions from mixed strategies, and incomplete information when the game is viewed from one player's position (*i.e.* hidden opponent cards). While an agent's performance can be estimated by simply averaging over a sufficiently large number of observed games, in many settings we would like to draw conclusions from as few games as possible. For example, in matches against human players, we may only be able to play a few thousand games. In an online setting, if we intend to adapt to the opponent, we would like to draw accurate conclusions in even shorter time scales in order to respond before the opponent changes their behaviour.

In this chapter's paper, we describe the Imaginary Observations technique for agent evaluation. This technique derives an unbiased, low-variance estimate of an agent's performance in a game, by leveraging our knowledge of the agent's strategy. Specifically, the technique examines each observed game, and then considers the much larger set of possible outcomes that would have appeared identical to the opponent, and thus elicited the same behaviour from them. Each "real" game is thus turned into thousands of "imaginary" games that can be averaged over, weighted by importance sampling, in order to derive a much lower variance estimate.

This technique has two key applications. First, it can be used as an accurate postgame analysis tool for evaluating games with complete information. In Chapter 8, we will demonstrate this by analyzing the 2007 and 2008 Man-vs-Machine Poker Championships with greater precision than any previously published analysis. Second, it can be used for online estimation, so that an agent can accurately estimate its performance during a match. The technique can also be used off-policy, allowing it to play strategy A and evaluate how well strategy B would have performed. This second application allows our agent to dynamically adapt during a match by switching between a set of candidate strategies in order to select the one with the highest expected value against the opponent. Our Polaris agent used this approach in the 2008 Man-vs-Machine Poker Championship to switch between five component strategies of varying aggressiveness, and thus adapt to and exploit its human adversaries.

DBR and Imaginary Observations now form two key steps of our current approach for opponent modelling and online adaptation, called the Implicit Modelling framework. Developed by Bard, Bowling, myself and our colleagues [8; 7; 9], this framework avoids "explicit models" in which we must learn how the opponent behaves at billions of decision points, and instead learns "implicit models" that describe the expected value of using each member of a set of precomputed strong strategies. In this framework, we anticipate the types of opponents that we might face and collect observations of their play, cluster the opponents into categories, use DBR to construct counter-strategies to each category's observations, and then use the Imaginary Observations technique online to switch between these counter-strategies. In this way, our agent combines the advantages of using offline computation to construct a strong, balanced set of strategies, with the ability to adapt to the opponent online in order to increase its expected winnings.

Author's contributions. The idea behind the Imaginary Observations technique was developed by Bowling. Bowling also contributed the theoretical foundation established throughout Section 7.3. I created our implementation of the algorithm and performed all of the experiments. All four authors were responsible for writing and editing the paper.

Strategy Evaluation in Extensive Games with Importance Sampling[1]

Michael Bowling (bowling@cs.ualberta.ca)
Michael Johanson (johanson@cs.ualberta.ca)
Neil Burch (burch@cs.ualberta.ca)
Duane Szafron (duane@cs.ualberta.ca)

Abstract: Typically agent evaluation is done through Monte Carlo estimation. However, stochastic agent decisions and stochastic outcomes can make this approach inefficient, requiring many samples for an accurate estimate. We present a new technique that can be used to simultaneously evaluate many strategies while playing a single strategy in the context of an extensive game. This technique is based on importance sampling, but utilizes two new mechanisms for significantly reducing variance in the estimates. We demonstrate its effectiveness in the domain of poker, where stochasticity makes traditional evaluation problematic.

7.1 Introduction

Evaluating an agent's performance is a component of nearly all research on sequential decision making. Typically, the agent's expected payoff is estimated through Monte Carlo samples of the (often stochastic) agent acting in an (often stochastic) environment. The degree of stochasticity in the environment or agent behavior determines how many samples are needed for an accurate estimate of performance. For results in synthetic domains with artificial agents, one can simply continue drawing samples until the estimate is accurate enough. For non-synthetic environments, domains that involve human participants, or when evaluation is part of an on-line algorithm, accurate estimates with a small number of samples are critical. This paper describes a new technique for tackling this problem in the context of extensive games.

An extensive game is a formal model of a sequential interaction between multiple, independent agents with imperfect information. It is a powerful yet compact framework for describing many strategic interactions between decision-makers, artificial and human[2]. Poker, for example, is a domain modeled very naturally as an extensive game. It involves independent and self-interested agents making sequential decisions based on both public and private information in a stochastic environment. Poker also demonstrates the challenge of evaluating agent performance. In one typical variant of poker, approximately 30,000 hands (or samples of playing the game) are sometimes needed to distinguish between professional and amateur

[1]The paper presented in this chapter originally appeared at the Twenty-Fifth International Conference on Machine Learning (ICML-08). M. Bowling, M. Johanson, N. Burch and D. Szafron. Strategy Evaluation in Extensive Games with Importance Sampling. Proceedings of the 25th International Conference on Machine Learning (ICML-08), 72-79, 2008.

[2]In this work we use the words "agent", "player", and "decision-maker" interchangeably and, unless explicitly stated, aren't concerned if they are humans or computers.

levels of play. Matches between computer and human opponents typically involve far fewer hands, yet still need to draw similar statistical conclusions.

In this work, we present a new technique for deriving low variance estimators of agent performance in extensive games. We employ importance sampling while exploiting the fact that the strategy of the agent being evaluated is typically known. However, we reduce the variance that importance sampling normally incurs by selectively adding synthetic data that is derived from but consistent with the sample data. As a result we derive low-variance unbiased estimators for agent performance given samples of the outcome of the game. We further show that we can efficiently evaluate one strategy while only observing samples from another. Finally, we examine the important case where we only get partial information of the game outcome (e.g., if a player folds in poker, their private cards are not revealed during the match and so the sequence of game states is not fully known). All of our estimators are then evaluated empirically in the domain of poker in both full and partial information scenarios.

This paper is organized as follows. In Section 7.2 we introduce the extensive game model, formalize our problem, and describe previous work on variance reduction in agent evaluation. In Section 7.3 we present a general procedure for deriving unbiased estimators and give four examples of these estimators. We then briefly introduce the domain of poker in Section 7.4 and describe how these estimators can be applied to this domain. In Section 7.5 we show empirical results of our approach in poker. Finally, we conclude in Section 7.6 with some directions for future work.

7.2 Background

We begin by describing extensive games and then we formalize the agent evaluation problem.

7.2.1 Extensive Games

Definition 4 *[76, p. 200] a finite extensive game with imperfect information has the following components:*

- *A finite set N of **players**.*
- *A finite set H of sequences, the possible **histories** of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the **terminal histories** (those which are not a prefix of any other sequences). $A(h) = \{a : (h, a) \in H\}$ are the actions available after a non-terminal history $h \in H$,*
- *A **player function** P that assigns to each non-terminal history (each member of $H \backslash Z$) a member of $N \cup \{c\}$, where c represents chance. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$, then chance determines the action taken after history h.*

- *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*
- *For each player $i \in N$ a partition $\mathbf{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. $\mathbf{I}_i$ is the **information partition** of player i; a set $I_i \in \mathbf{I}_i$ is an **information set** of player i.*
- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals $\mathbf{R}$. If $N = \{1, 2\}$ and $u_1 = -u_2$, it is a **zero-sum extensive game**.*

A **strategy of player *i*** σ_i in an extensive game is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathbf{I}_i$. A **strategy profile** σ consists of a strategy for each player, $\sigma_1, \sigma_2, \ldots$, with σ_{-i} referring to all the strategies in σ except σ_i.

Let $\pi^\sigma(h)$ be the probability of history h occurring if players choose actions according to σ. We can decompose $\pi^\sigma = \Pi_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ into each player's contribution to this probability. Hence, $\pi_i^\sigma(h)$ is the probability that if player i plays according to σ then for all histories h' that are a proper prefix of h with $P(h') = i$, player i takes the subsequent action in h. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except player i. The overall value to player i of a strategy profile is then the expected payoff of the resulting terminal node, i.e., $u_i(\sigma) = \sum_{z \in Z} u_i(z)\pi^\sigma(z)$. For $Y \subseteq Z$, a subset of possible terminal histories, define $\pi^\sigma(Y) = \sum_{z \in Y} \pi^\sigma(z)$, to be the probability of reaching any outcome in the set Y given σ, with $\pi_i^\sigma(Y)$ and $\pi_{-i}^\sigma(Y)$ defined similarly.

7.2.2 The Problem

Given some function on terminal histories $V : Z \to \Re$ we want to estimate $E_{z|\sigma}[V(z)]$. In most cases V is simply u_i, and the goal is to evaluate a particular player's expected payoff. We explore three different settings for this problem. In all three settings, we assume that σ_i (our player's strategy) is known, while $\sigma_{j \neq i}$ (the other players' strategies) are not known.

- *On-policy full-information*. In the simplest case, we get samples $z_{1\ldots t} \in Z$ from the distribution π^σ.
- *Off-policy full-information*. In this case, we get samples $z_{1\ldots t} \in Z$ from the distribution $\pi^{\hat{\sigma}}$ where $\hat{\sigma}$ differs from σ only in player i's strategy: $\pi_{-i}^\sigma = \pi_{-i}^{\hat{\sigma}}$. In this case we want to evaluate one strategy for player i from samples of playing a different one.
- *Off-policy partial-information*. In the hardest case, we don't get full samples of outcomes z_t, but rather just player i's view of the outcomes. For example, in poker, if a player folds, their cards are not revealed to the other players and so certain chance actions are not known. Formally, in this case we get

samples of $K(z_t) \in \mathbf{K}$, where K is a many-to-one mapping and z_t comes from the distribution $\pi^{\hat{\sigma}}$ as above. K intuitively must satisfy the following conditions: for $z, z' \in Z$, if $K(z) = K(z')$ then,

- $V(z) = V(z')$, and
- $\forall \sigma \quad \pi_i^\sigma(z) = \pi_i^\sigma(z')$.

7.2.3 Monte Carlo Estimation

The typical approach to estimating $E_{z|\sigma}[V(z)]$ is through simple Monte Carlo estimation. Given independent samples $z_1, \ldots, z_t$ from the distribution π^σ, simply estimate the expectation as the sample mean of outcome values.

$$\frac{1}{t} \sum_{i=1}^{t} V(z_i) \tag{7.1}$$

As the estimator has zero bias, the mean squared error of the estimator is determined by its variance. If the variance of $V(z)$ given σ is large, the error in the estimate can be large and many samples are needed for accurate estimation.

Recently, we proposed a new technique for agent evaluation in extensive games [107]. We showed that value functions over non-terminal histories could be used to derive alternative unbiased estimators. If the chosen value function was close to the true expected value given the partial history and players' strategies, then the estimator would result in a reduction in variance. The approach essentially derives a real-valued function $\tilde{V}(z)$ that is used in place of V in the Monte Carlo estimator from Equation 7.1. The expectation of $\tilde{V}(z)$ matches the expectation of $V(z)$ for any choice of σ, and so the result is an unbiased estimator, but potentially with lower variance and thus lower mean-squared error. The specific application of this approach to poker, using an expert-defined value function, was named the DIVAT estimator and was shown to result in a dramatic reduction in variance. A simpler choice of value function, the expected value assuming the betting is "bet-call" for all remaining betting rounds, can even make a notable reduction. We refer to this conceptually and computationally simpler estimator as (Bet-Call) BC-DIVAT.

Both traditional Monte Carlo estimation and DIVAT are focused on the *on-policy* case, requiring outcomes sampled from the joint strategy that is being evaluated. Furthermore, DIVAT is restricted to *full-information*, where the exact outcome is known. Although limited in these regards, they also don't require any knowledge about any of the players' strategies.

7.3 General Approach

We now describe our new approach for deriving low-variance, unbiased estimators for agent evaluation. In this section we almost exclusively focus on the *off-policy full-information* case. Within this setting we observe a sampled outcome z from the

distribution $\pi^{\hat{\sigma}}$, and the goal is to estimate $E_{z|\sigma}[V(z)]$. The outcomes are observed based on the strategy $\hat{\sigma}$ while we want to evaluate the expectation over σ, where they differ only in player i's strategy. This case subsumes the on-policy case, and we touch on the more difficult partial-information case at the end of this section. In order to handle this more challenging case, we require full knowledge of player i's strategies, both the strategy being observed $\hat{\sigma}_i$ and the one being evaluated σ_i.

At the core of our technique is the idea that synthetic histories derived from the sampled history can also be used in the estimator. For example, consider the unlikely case when σ is known entirely. Given an observed outcome $z \in Z$ (or even without an observed outcome) we can exactly compute the desired expectation by examining every outcome.

$$V_Z(z) \equiv \sum_{z' \in Z} V(z')\pi^{\sigma}(z') = E_{z|\sigma}[V(z)] \tag{7.2}$$

Although impractical since we don't know σ, $V_Z(z)$ is an unbiased and zero variance estimator.

Instead of using every terminal history, we could restrict ourselves to a smaller set of terminal histories. Let $U(z' \in Z) \subseteq Z$ be a mapping of terminal histories to a set of terminal histories, where at least $z' \in U(z')$. We can construct an unbiased estimator that considers the history z' in the estimation whenever we observe a history from the set $U(z')$. Another way to consider things is to say that $U^{-1}(z)$ is the set of synthetic histories considered when we observe z. Specifically, we define the estimator $V_U(z)$ for the observed outcome z as,

$$V_U(z) \equiv \sum_{z' \in U^{-1}(z)} V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} \tag{7.3}$$

The estimator considers the value of every outcome z' where the observed history z is in the set $U(z')$. Each outcome though is weighted in a fashion akin to importance sampling. The weight term for z' is proportional to the probability of that history given σ, and inversely proportional to the probability that z' is one of the considered synthetic histories when observing sampled outcomes from $\hat{\sigma}$. Note that $V_U(z)$ is not an estimate of $V(z)$, but rather has the same expectation.

At first glance, V_U may seem just as impractical as V_Z since σ is not known. However, with a careful choice of U we can insure that the weight term depends only on the known strategies σ_i and $\hat{\sigma}_i$. Before presenting example choices of U, we first prove that V_U is unbiased.

Theorem 9 *If $\pi_i^{\hat{\sigma}}(z)$ is non-zero for all outcomes $z \in Z$, then,*

$$E_{z|\hat{\sigma}}[V_U(z)] = E_{z|\sigma}[V(z)],$$

i.e., V_U is an unbiased estimator.

Proof First, let us consider the denominator in the weight term of V_U. Since $z' \in U(z')$ and $\pi_i^{\hat{\sigma}}$ is always positive, the denominator can only be zero if $\pi_{-i}^{\hat{\sigma}}(z')$ is zero. If this were true, $\pi_{-i}^{\sigma}(z')$ must also be zero, and as a consequence so must the numerator. As a result the terminal history z' is never reached and so it is correct to simply exclude such histories from the estimator's summation.

Define $\mathbf{1}(x)$ to be the indicator function that takes on the value 1 if x is true and 0 if false.

$$
\begin{aligned}
&E_{z|\hat{\sigma}}\left[V_U(z)\right] \\
&= E_{z|\hat{\sigma}}\left[\sum_{z' \in U^{-1}(z)} V(z') \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}\right] && (7.4)\\
&= E_{z|\hat{\sigma}}\left[\sum_{z'} \mathbf{1}(z \in U(z')) V(z') \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}\right] && (7.5)\\
&= \sum_{z'} V(z') \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} E_{z|\hat{\sigma}}\left[\mathbf{1}(z \in U(z'))\right] && (7.6)\\
&= \sum_{z'} V(z') \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} \pi^{\hat{\sigma}}(U(z')) && (7.7)\\
&= \sum_{z'} V(z') \pi^{\sigma}(z') = E_{z|\sigma}\left[V(z)\right] && (7.8)
\end{aligned}
$$

The derivation follows from the linearity of expectation, the definition of $\pi^{\hat{\sigma}}$, and the definition of expectation. □

We now look at four specific choices of U for which the weight term can be computed while only knowing player i's portion of the joint strategy σ.

Example 1: Basic Importance Sampling. The simplest choice of U for which V_U can be computed is $U(z) = \{z\}$. In other words, the estimator considers just the sampled history. In this case the weight term is:

$$
\begin{aligned}
\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} &= \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(z')} && (7.9)\\
&= \frac{\pi_i^{\sigma}(z')\pi_{-i}^{\sigma}(z')}{\pi_i^{\hat{\sigma}}(z')\pi_{-i}^{\hat{\sigma}}(z')} && (7.10)\\
&= \frac{\pi_i^{\sigma}(z')}{\pi_i^{\hat{\sigma}}(z')} && (7.11)
\end{aligned}
$$

The weight term only depends on σ_i and $\hat{\sigma}_i$ and so is a known quantity. When $\hat{\sigma}_i = \sigma_i$ the weight term is 1 and the result is simple Monte Carlo estimation. When $\hat{\sigma}_i$ is different, the estimator is a straightforward application of importance sampling.

Example 2: Game Ending Actions. A more interesting example is to consider all histories that differ from the sample history by only a single action by player i and that action must be the last action in the history. For example, in poker, the history where the player being evaluated chooses to fold at an earlier point in the betting sequence is considered in this estimator. Formally, define $S_{-i}(z) \in H$ to be the shortest prefix of z where the remaining actions in z are all made by player i or chance. Let $U(z) = \{z' \in Z : S_{-i}(z) \text{ is a prefix of } z'\}$. The weight term becomes,

$$\frac{\pi^\sigma(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^\sigma(z')}{\pi^{\hat{\sigma}}(S_{-i}(z'))} \tag{7.12}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z'))\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{7.13}$$

$$= \frac{\pi^\sigma_{-i}(S_{-i}(z'))\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z'))\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{7.14}$$

$$= \frac{\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{7.15}$$

As this only depends on the strategies of player i, we can compute this quantity and therefore the estimator.

Example 3: Private Information. We can also use all histories in the update that differ only in player i's private information. In other words, any history that the other players wouldn't be able to distinguish from the sampled history is considered. For example, in poker, any history where player i receiving different private cards is considered in the estimator since the opponents' strategy cannot depend directly on this strictly private information. Formally, let $U(z) = \left\{z' \in Z : \forall \sigma\ \pi^\sigma_{-i}(z') = \pi^\sigma_{-i}(z)\right\}$. The weight term then becomes,

$$\frac{\pi^\sigma(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^\sigma(z')}{\sum_{z'' \in U(z')} \pi^{\hat{\sigma}}(z'')} \tag{7.16}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\sum_{z'' \in U(z')} \pi^{\hat{\sigma}}_{-i}(z'')\pi^{\hat{\sigma}}_i(z'')} \tag{7.17}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\sum_{z'' \in U(z')} \pi^{\hat{\sigma}}_{-i}(z')\pi^{\hat{\sigma}}_i(z'')} \tag{7.18}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_{-i}(z')\sum_{z'' \in U(z')} \pi^{\hat{\sigma}}_i(z'')} \tag{7.19}$$

$$= \frac{\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_i(U(z'))} \tag{7.20}$$

As this only depends on the strategies of player i, we can again compute this quantity and therefore the estimator as well.

Example 4: Combined. The past two examples show that we can consider histories that differ in the player's private information or by the player making an alternative game ending action. We can also combine these two ideas and consider any

history that differs by both an alternative game ending action and the player's private information. Define $Q(z) = \{h \in H : |h| = |S_{-i}(z)| \text{ and } \forall \sigma \pi^{\sigma}_{-i}(h) = \pi^{\sigma}_{-i}(S_{-i}(z))\}$, Let $U(z) = \{z' \in Z : \text{ a prefix of } z' \text{ is in } Q(z)\}$.

$$\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(Q(z'))} \tag{7.21}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_{i}(z')}{\sum_{h \in Q(z')} \pi^{\hat{\sigma}}_{-i}(h)\pi^{\hat{\sigma}}_{i}(h)} \tag{7.22}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_{i}(z')}{\sum_{h \in Q(z')} \pi^{\hat{\sigma}}_{-i}(S_{-i}(z))\pi^{\hat{\sigma}}_{i}(h)} \tag{7.23}$$

$$= \frac{\pi^{\sigma}_{-i}(S_{-i}(z'))\pi^{\sigma}_{i}(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z'))\sum_{h \in Q(z')} \pi^{\hat{\sigma}}_{i}(h)} \tag{7.24}$$

$$= \frac{\pi^{\sigma}_{i}(z')}{\pi^{\hat{\sigma}}_{i}(Q(z'))} \tag{7.25}$$

Once again this quantity only depends on the strategies of player i and so we can compute this estimator as well.

We have presented four different estimators that try to extract additional information from a single observed game outcome. We can actually combine any of these estimators with other unbiased approaches for reducing variance. This can be done by replacing the V function in the above estimators with any unbiased estimate of V. In particular, these estimators can be combined with our previous DIVAT approach by choosing V to be the DIVAT (or BC-DIVAT) estimator instead of u_i.

7.3.1 Partial Information

The estimators above are provably unbiased for both the on-policy and off-policy full-information case. We now briefly discuss the off-policy partial-information case. In this case we don't directly observe the actual terminal history z_t but only a many-to-one mapping $K(z_t)$ of the history. One simple adaptation of our estimators to this case is to use the history z' in the estimator whenever it is possible that the unknown terminal history could be in $U(z')$, while keeping the weight term unchanged. Although we lose the unbiased guarantee with these estimators, it is possible that the reduction in variance is more substantial than the error caused by the bias. We investigate empirically the magnitude of the bias and the resulting mean-squared error of such estimators in the domain of poker in Section 7.5.

7.4 Application to Poker

To analyze the effectiveness of these estimators, we will use the popular game of Texas Hold'em poker, as played in the AAAI Computer Poker Competition [65]. The game is two-player and zero-sum. Private cards are dealt to the players, and

over four rounds, public cards are revealed. During each round, the players place bets that the combination of their public and private cards will be the strongest at the end of the game. The game has just under 10^{18} game states, and has the properties of imperfect information, stochastic outcomes, and observations of the game outcome during a match exhibit partial information.

Each of the situations described in Section 7.2, on-policy and off-policy as well as full-information and partial information, have relevance in the domain of poker. In particular, the *on-policy full-information* case is the situation where one is trying to evaluate a strategy from full-information descriptions of the hands, as might be available after a match is complete. For example, this could be used to more accurately determine the winner of a competition involving a small number of hands (which is always the case when humans are involved). In this situation it is critical, that the estimator is unbiased, i.e., it is an accurate reflection of the expected winnings and therefore does not incorrectly favor any playing style.

The *off-policy full-information* case is useful for examining past games against an opponent to determine which of many alternative strategies one might want to use against them in the future. The introduction of bias (depending on the strategy used when playing the past hands) is not problematic, as the goal in this case is an estimate with as little error as possible. Hence the introduction of bias is acceptable in exchange for significant decreases in variance.

Finally, the *off-policy partial-information* case corresponds to evaluating alternative strategies during an actual match. In this case, we want to evaluate a set of strategies, which aren't being played, to try and identify an effective choice for the current opponent. The player could then choose a strategy whose performance is estimated to be strong even for hands it wasn't playing.

The estimators from the previous section all have natural applications to the game of poker:

- **Basic Importance Sampling**. This is a straightforward application of importance sampling. The value of the observed outcome of the hand is weighted by the ratio of the probability that the strategy being evaluated (σ_i) takes the same sequence of actions to the probability that the playing strategy ($\hat{\sigma}_i$) takes the sequence of actions.

- **Game ending actions**. By selecting the *fold* betting action, a player surrenders the game in order to avoid matching an opponent's bet. Therefore, the game ending actions estimator can consider all histories in which the player could have folded during the observed history.[3] We call this the **Early Folds** (EF) estimator. The estimator sums over all possible prefixes of the betting sequence where the player could have chosen to fold. In the summation it weights the value of surrendering the pot at that point by the ratio of the probability of the observed betting up to that point and then folding given the

[3] In the full-information setting we can also consider situations where the player could have *called* on the final round of betting to end the hand.

player's cards (and σ_i) to the probability of the observed betting up to that point given the player's cards (and $\hat{\sigma}_i$).

- **Private information**. In Texas Hold'em, a player's private information is simply the two private cards they are dealt. Therefore, the private information estimator can consider all histories with the same betting sequence in which the player holds different private cards. We call this the **All Cards** (AC) estimator. The estimator sums over all possible two-card combinations (excepting those involving exposed board or opponent cards). In the summation it weights the value of the observed betting with the imagined cards by the ratio of the probability of the observed betting given those cards (and σ_i) to the probability of the observed betting (given $\hat{\sigma}_i$) summed over all cards.

7.5 Results

Over the past few years we have created a number of strong Texas Hold'em poker agents that have competed in the past two AAAI Computer Poker Competitions. To evaluate our new estimators, we consider games played between three of these poker agents: S2298 [108], PsOpti4 [11], and CFR8 [110]. In addition, we also consider Orange, a competitor in the First Man-Machine Poker Championship.

To evaluate these estimators, we examined records of games played between each of three candidate strategies (S2298, CFR8, Orange) against the opponent PsOpti4. Each of these three records contains one million hands of poker, and can be viewed as full information (both players' private cards are always shown) or as partial information (when the opponent folds, their private cards are not revealed). We begin with the full-information experiments.

7.5.1 Full Information

We used the estimators described previously to find the value of each of the three candidate strategies, using full-information records of games played from just one of the candidate strategies. The strategy that actually played the hands in the record of games is called the on-policy strategy and the others are the off-policy strategies. The results of one these experiments is presented in Table 7.1. In this experiment, we examined one million full-information hands of S2298 playing against PsOpti4. S2298 (the on-policy strategy) and CFR8 and Orange (the off-policy strategies) are evaluated by our importance sampling estimators, as well as DIVAT, BC-DIVAT, and a few combination estimators. We present the empirical bias and standard deviation of the estimators in the first two columns. The third column, "RMSE", is the root-mean-squared error of the estimator if it were used as the method of evaluation for a 1000 hand match (a typical match length). All of the numbers are reported in millibets per hand played. A millibet is one thousandth of a small-bet, the fixed magnitude of bets used in the first two rounds of betting. To provide some intuition for these numbers, a player that always folds will lose 750 millibets per

	Bias	StdDev	RMSE
S2298			
Basic	0*	5103	161
DIVAT	0*	1935	61
BC-DIVAT	0*	2891	91
Early Folds	0*	5126	162
All Cards	0*	4213	133
AC+BC-DIVAT	0*	2146	68
AC+EF+BC-DIVAT	0*	1778	56
CFR8			
Basic	200 ± 122	62543	1988
DIVAT	62 ± 104	53033	1678
BC-DIVAT	84 ± 45	22303	710
Early Folds	123 ± 120	61481	1948
All Cards	12 ± 16	8518	270
AC+BC-DIVAT	35 ± 13	3254	109
AC+EF+BC-DIVAT	2 ± 12	2514	80
Orange			
Basic	159 ± 40	20559	669
DIVAT	3 ± 25	11350	359
BC-DIVAT	103 ± 28	12862	420
Early Folds	82 ± 35	17923	572
All Cards	7 ± 16	8591	272
AC+BC-DIVAT	8 ± 13	3154	100
AC+EF+BC-DIVAT	6 ± 12	2421	77

Table 7.1: *Full Information Case.* Empirical bias, standard deviation, and root mean-squared-error over a 1000 hand match for various estimators. 1 million hands of poker between S2298 and PsOpti4 were observed. A bias of 0* indicates a provably unbiased estimator.

hand, and strong players aim to achieve an expected win rate over 50 millibets per hand.

In the on-policy case, where we are evaluating S2298, all of the estimators are provably unbiased, and so they only differ in variance. Note that the Basic estimator, in this case, is just the Monte-Carlo estimator over the actual money lost or won. The Early Folds estimator provides no variance reduction over the Monte-Carlo estimate, while the All Cards estimator provides only a slight reduction. However, this is not nearly as dramatic as the reduction provided by the DIVAT estimator. The importance sampling estimators, however, can be combined with the DIVAT estimator as described in Section . The combination of BC-DIVAT with All Cards ("AC+BC-DIVAT") results in lower variance than either of the estimators separately.[4] The

[4]The importance sampling estimators were combined with BC-DIVAT instead of DIVAT because the original DIVAT estimator is computationally burdensome, particularly when many evaluations

addition of Early Folds ("AC+EF+BC-DIVAT") produces an even further reduction in variance, showing the best-performance of all the estimators, even though Early Folds on its own had little effect.

In the off-policy case, where we are evaluating CFR8 or Orange, we report the empirical bias (along with a 95% confidence bound) in addition to the variance. As DIVAT and BC-DIVAT were not designed for off-policy evaluation, we report numbers by combining them with the Basic estimator (i.e., using traditional importance sampling). Note that bias is possible in this case because our on-policy strategy (S2298) does not satisfy the assumption in Theorem 9, as there are some outcomes the strategy never plays. Basic importance sampling in this setting not only shows statistically significant bias, but also exhibits impractically large variance. DIVAT and BC-DIVAT, which caused considerable variance reduction on-policy, also should considerable variance reduction off-policy, but not enough to offset the extra variance from basic importance sampling. The All Cards estimator, on the other hand, shows dramatically lower variance with very little bias (in fact, the empirical bias is statistically insignificant). Combining the All Cards estimator with BC-DIVAT and Early Folds further reduces the variance, giving off-policy estimators that are almost as accurate as our best on-policy estimators.

The trends noted above continue in the other experiments, when CFR8 and Orange are being observed. For space considerations, we don't present the individual tables, but instead summarize these experiments in Table 7.2. The table shows the minimum and maximum empirically observed bias, standard deviation, and the root-mean-squared error of the estimator for a 1000 hand match. The strategies being evaluated are separated into the on-policy case, when the record involves data from that strategy, and the off-policy case, when it doesn't.

7.5.2 Partial Information

The same experiments were repeated for the case of partial information. The results of the experiment involving S2298 playing against PsOpti4 and evaluating our three candidate strategies under partial information is shown in Table 7.3. For DIVAT and BC-DIVAT, which require full information of the game outcome, we used a partial information variant where the full-information estimator was used when the game outcome was known (i.e., no player folded) and winnings was used when it was not. This variant can result in a biased estimator, as can be seen in the table of results. The All Cards estimator, although also without any guarantee of being unbiased, actually fares much better in practice, not displaying a statistically significant bias in either the off-policy or on-policy experiments. However, even though the DIVAT estimators are biased their low variance makes them preferred in terms of RMSE in the on-policy setting. In the off-policy setting, the variance caused by Basic importance sampling (as used with DIVAT and BC-DIVAT) makes the All Cards estimator the only practical choice. As in the full-information case we can combine the All Cards and BC-DIVAT for further variance reduction. The resulting estimator

are needed for every observation as is the case with the All Cards estimator.

	Bias			StdDev			RMSE		
	Min	–	Max	Min	–	Max	Min	–	Max
On Policy									
Basic	0*	–	0*	5102	–	5385	161	–	170
DIVAT	0*	–	0*	1935	–	2011	61	–	64
BC-DIVAT	0*	–	0*	2891	–	2930	91	–	92
AC+GE+BC-DIVAT	0*	–	0*	1701	–	1778	54	–	56
Off Policy									
Basic	49	–	200	20559	–	244469	669	–	7732
DIVAT	2	–	62	11350	–	138834	358	–	4390
BC-DIVAT	10	–	103	12862	–	173715	419	–	5493
AC+GE+BC-DIVAT	2	–	9	1816	–	2857	58	–	90

Table 7.2: *Summary of the Full-Information Case.* Summary of empirical bias, standard deviation, and root-mean-squared error over a 1000 hand match for various estimators. The minimum and maximum encountered values for all combinations of observed and evaluated strategies is presented. A bias of 0* indicates a provably unbiased estimator.

has lower RMSE than either All Cards or BC-DIVAT alone both in the on-policy and off-policy cases. The summary of the results of the other experiments, showing similar trends, are shown in Table 7.4.

7.6 Conclusion

We introduced a new method for estimating agent performance in extensive games based on importance sampling. The technique exploits the fact that the agent's strategy is typically known to derive several low variance estimators that can simultaneously evaluate many strategies while playing a single strategy. We prove that these estimators are unbiased in the on-policy case and (under usual assumptions) in the off-policy case. We empirically evaluate the techniques in the domain of poker, showing significant improvements in terms of lower variance and lower bias. We show that the estimators can also be used even in the challenging problem of estimation with partial information observations.

Acknowledgments

We would like to thank Martin Zinkevich and Morgan Kan along with all of the members of the University of Alberta Computer Poker Research Group for their valuable insights. This research was supported by NSERC and iCore.

	Bias	StdDev	RMSE
S2298			
Basic	0*	5104	161
DIVAT	81±9	2762	119
BC-DIVAT	95±9	2759	129
Early Folds	47±1	5065	167
All Cards	5±13	4218	133
AC+BC-DIVAT	96±12	2650	127
CFR8			
Basic	202±80	40903	1309
DIVAT	175±47	23376	760
BC-DIVAT	183±47	23402	762
Early Folds	181±78	39877	1274
All Cards	13±19	7904	250
AC+BC-DIVAT	101±16	4014	162
Orange			
Basic	204±45	23314	765
DIVAT	218±22	10029	385
BC-DIVAT	244±21	10045	401
Early Folds	218±43	22379	741
All Cards	3±19	8092	256
AC+BC-DIVAT	203±16	3880	237

Table 7.3: *Partial-Information Case.* Empirical bias, standard deviation, and root mean-squared-error over a 1000 hand match for various estimators. 1 million hands of poker between S2298 and PsOpti4 with partial information were observed. A bias of 0* indicates a provably unbiased estimator.

	Bias			StdDev			RMSE		
	Min	–	Max	Min	–	Max	Min	–	Max
On Policy									
Basic	0*	–	0*	5104	–	5391	161	–	170
DIVAT	56	–	144	2762	–	2876	105	–	170
BC-DIVAT	78	–	199	2759	–	2859	118	–	219
AC+BC-DIVAT	78	–	206	2656	–	2766	115	–	224
Off Policy									
Basic	17	–	433	23314	–	238874	753	–	7566
DIVAT	103	–	282	10029	–	88791	384	–	2822
BC-DIVAT	35	–	243	10045	–	99287	400	–	3139
AC+BC-DIVAT	63	–	230	3055	–	6785	143	–	258

Table 7.4: *Summary of the Partial-Information Case.* Summary of empirical bias, standard deviation, and root-mean-squared error over a 1000 hand match for various estimators. The minimum and maximum encountered values for all combinations of observed and evaluated strategies is presented. A bias of 0* indicates a provably unbiased estimator.

Chapter 8

Man-vs-Machine Poker Championships

The Abstraction-Solving-Translation procedure that these papers have described provides a practical approach for creating effective strategies and counter-strategies. In addition to the objective evaluation provided by best response calculations, we have also empirically validated the agents we have created through comparisons against computer and human opponents. For example, in the 45 ACPC events held between 2006 and 2014, the University of Alberta has taken first place 26 times and second or third an additional 17 times, failing to medal in only two events. Many of the top computer agents in the ACPC now use the abstraction and game solving algorithms that we have pioneered: of the top three agents in each event, 13 out of 18 in 2013 and 10 out of 18 in 2014 used CFR and abstractions derived from the techniques presented in Chapter 5. Further, several of our colleagues such as Oskari Tammelin, Eric Jackson, Mihai Ciucu, and Sandholm, Ganzfried, and Brown at Carnegie Mellon University have extended our algorithms such as CFR and defeated us in ACPC events.

Competitions against top human experts provide another empirical evaluation, in which we can compare artificial intelligence against human intelligence. The moment at which computer agents first surpass human abilities in a game serves as a milestone event in the public's understanding of artificial intelligence, such as the checkers-playing program Chinook becoming the first computer agent to win a world championship [86], IBM's Deep Blue achieving its famous victory over Kasparov in chess [43], and IBM's Watson defeating Jennings and Rutter on Jeopardy! [29]. Once computer agents surpass human expertise in a game, the second – and final – milestone to achieve is when the game is solved, such that a computer agent will never lose again against any adversary.

My research that has been presented in this thesis has played a central role in meeting both of these milestones in the poker domain, in the game of heads-up limit Texas hold'em. In this chapter, we will focus on the first milestone of defeating top human professionals, which occurred in 2008 at the beginning of my doctorate. In the next chapter, we will focus on the second milestone of solving the game, which occurred in January 2015 at the end of my doctorate.

In 2007 and 2008, the University of Alberta held two Man-vs-Machine Poker Championships, using the game of heads-up limit Texas hold'em. In each event our agent, Polaris, competed in a series of duplicate poker matches against teams of human professionals.[1] Polaris narrowly lost the 2007 match against Phil Laak and Ali Eslami, but demonstrated that it was approaching professional calibre play. In the 2008 match, an improved version of Polaris played a against a team of heads-up limit Texas hold'em specialists and narrowly won, marking the first time that a computer had defeated expert human poker players in a meaningful competition.

Although the variance-reducing duplicate format was used in the 2007 and 2008 matches, the matches still had enough variance that the results were not statistically significant. In my Masters thesis, published just after the 2007 match, I presented a further analysis using the DIVAT variance reduction technique [107; 14] that predicted Polaris should have won the 2007 match, but again without statistical significance [47, Section 7.3]. Likewise, an analysis of the 2008 match using DIVAT helps, but does not provide statistically significant results.

In this chapter, we will re-examine the 2007 and 2008 Man-vs-Machine Poker Championships. Using the Imaginary Observations technique that we presented in Chapter 7, we can now analyze the matches hand-by-hand to produce an unbiased low-variance evaluation of Polaris' performance. This analysis, previously unpublished and appearing here for the first time, predicts with 95% confidence that Polaris was favoured to win the 2007 event, and that Polaris earned its victory in the 2008 event.

8.1 The First Man-vs-Machine Poker Championship, 2007

The First Man-vs-Machine Poker Championship was held in July 2007 during the AAAI Conference on Artificial Intelligence.[2] Our opponents in this event were Phil Laak, a well-known professional poker player, and his chosen teammate Ali Eslami. The two-day event used a series of four 500-hand duplicate matches (4000 hands total), in which Laak and Eslami separately played against independent copies of Polaris. In each match, one human player played on a stage in front of an audience of AAAI attendees, while their partner played while sequestered in a hotel room. In between matches, the players were free to meet to exchange opinions and plan their strategy for the next match; likewise, the CPRG had an opportunity to adjust Polaris in between matches, and the program played autonomously during each match.

[1] In a duplicate poker match, two teams simultaneously play independent matches in different rooms. The same cards are dealt in each match, with the players acting in opposite positions in each room, or "side of the match". This format reduces the impact of luck, because when a player is dealt strong cards in one room, their teammate's opponent is dealt the same strong cards in the other room. In a duplicate match against a computer program, two human players act as a team, and each plays against an independent copy of the program.

[2] See [74] for the website used during the event, which includes photos, media links, and a minute-by-minute commentary of each match.

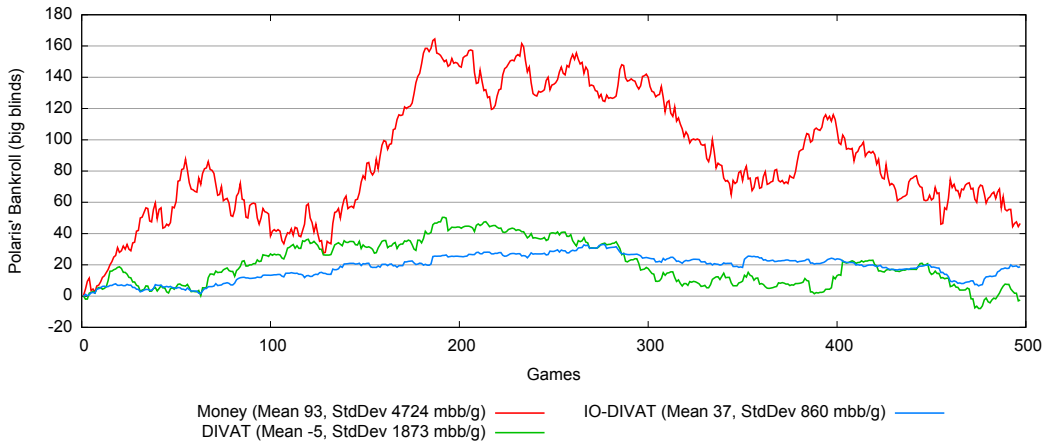
Using the duplicate format the same cards were dealt in each room, with the humans playing in opposite positions, such that both Laak and the Polaris agent opposing Eslami were dealt the same cards. The duplicate format helped to reduce the variance of the game[3], but with 4000 hands was still not expected to be sufficient to declare a winner with statistical significance. Instead, the rules defined a margin of victory: in each 500-hand match, a winner was declared if their team's score exceeded 25 big blinds. The overall winner of the Championship would be the team that won the most matches.

Polaris used a set of strategies computed by the newly-invented CFR algorithm, and used perfect recall percentile hand strength card abstractions. These abstractions used just 8 or 12 buckets per round, and were both tiny and crude by modern standards, as compared to the techniques we presented in Chapter 5. Polaris used a slightly different strategy in each of the four matches, and in the third match switched between several strategies in an attempt to exploit the opponent's weaknesses. Our agreement with the players was that each of its strategies would be given an arbitrary name, which we would tell them at the start of each match. Thus, if Polaris used the same strategy in two matches, the humans would know. For detailed information on the specific strategies used in each match, consult my Masters thesis [47, Section 7.3], which was written just after the competition.

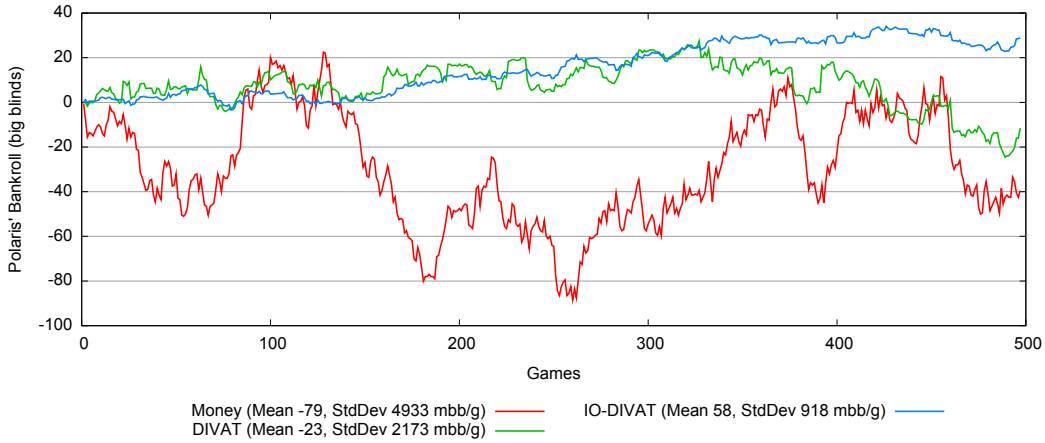
In the competition, the raw "Money" outcome of each duplicate match was used to choose a winner for the match, but these selections were not statistically significant. A further analysis in my Masters thesis [47, Section 7.3] using the DIVAT technique [107; 14] suggested that Polaris had an edge, but also could not identify a winner of each match with 95% confidence.

Now, using the combination of the Imaginary Observations technique from Chapter 7 and DIVAT, we can produce a more accurate value estimation for Polaris that achieves 95% confidence. These results are shown in Figures 8.1 to 8.4, which show Polaris's winnings or losses over the course of each match. Each set of three plots shows Polaris' performance against Laak, Eslami, and the Duplicate match that combines both opponents. The "Money" curve in each graph shows Polaris' actual winnings or losses, while DIVAT and IO-DIVAT show our variance-reduced estimates.

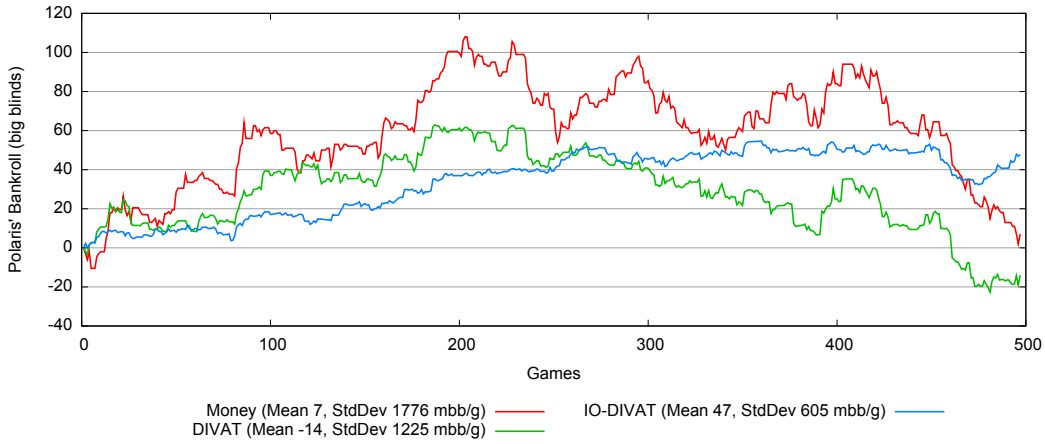
[3] In practice, we have found that heads-up limit Texas hold'em played between two strong players has a standard deviation of about 5000 mbb/g. In a duplicate match where the outcome of each game is the average of the two teammates' scores, in practice the standard deviation is about 1800 mbb/g.



(a) Phil Laak: Hotel Room

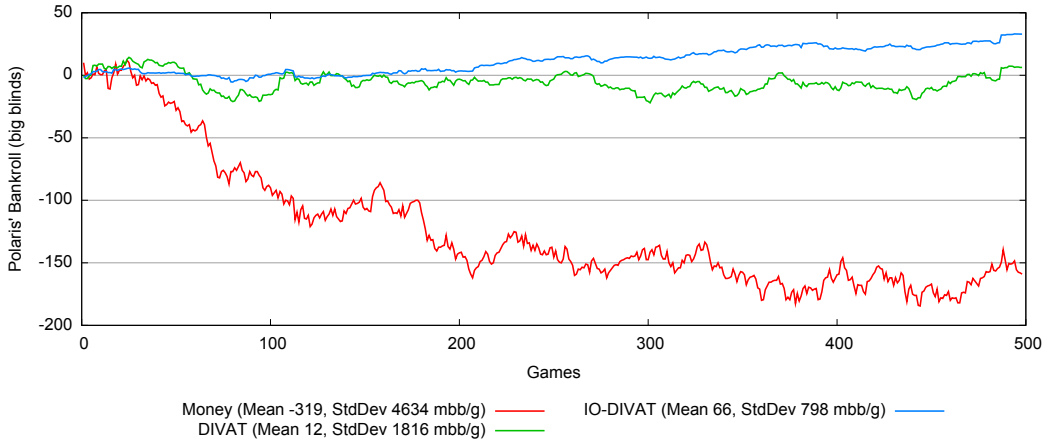


(b) Ali Eslami: On Stage

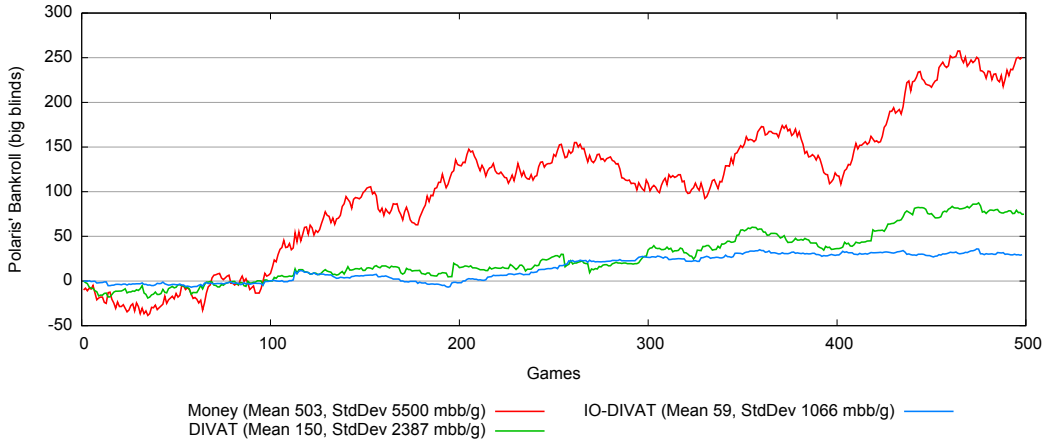


(c) Duplicate

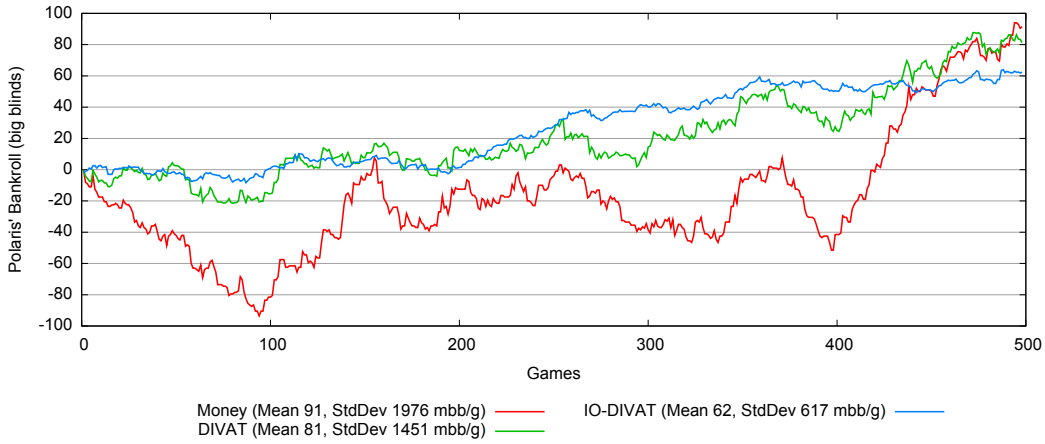
Figure 8.1: 2007 Man-vs-Machine Championship Match 1.



(a) Phil Laak: On Stage

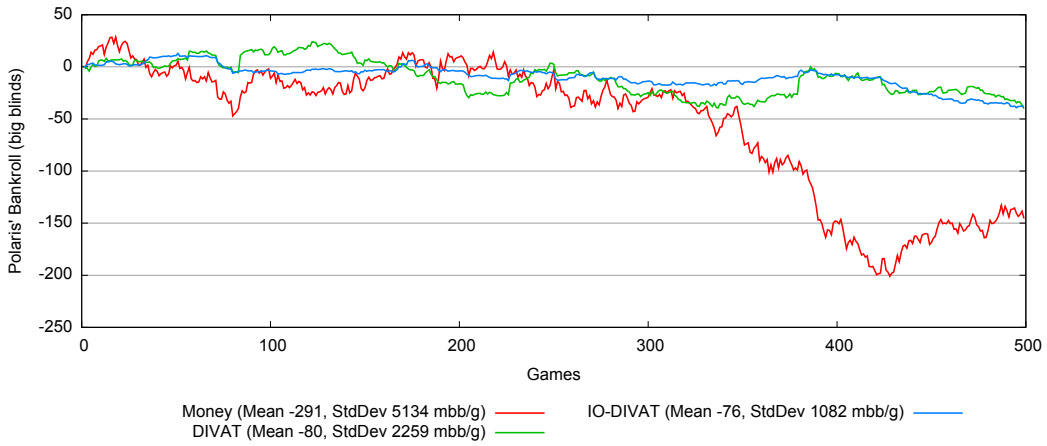


(b) Ali Eslami: Hotel Room

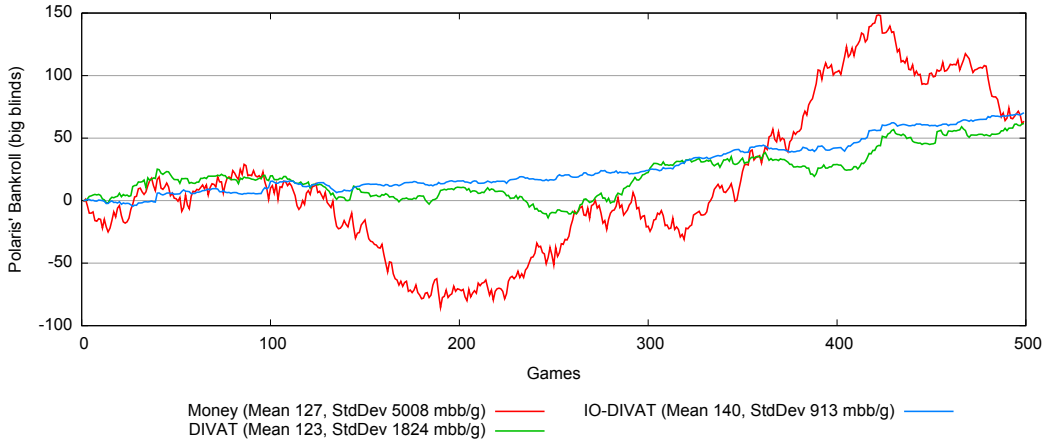


(c) Duplicate

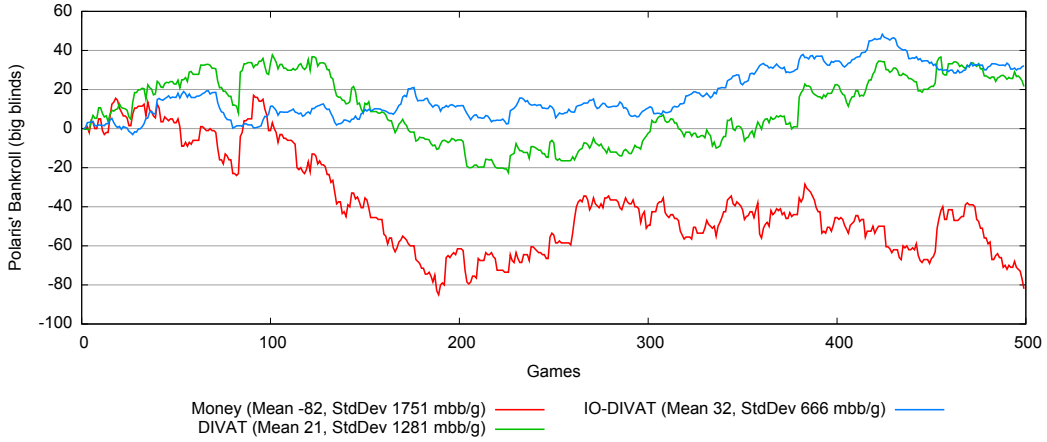
Figure 8.2: 2007 Man-vs-Machine Championship Match 2.



(a) Phil Laak: On Stage

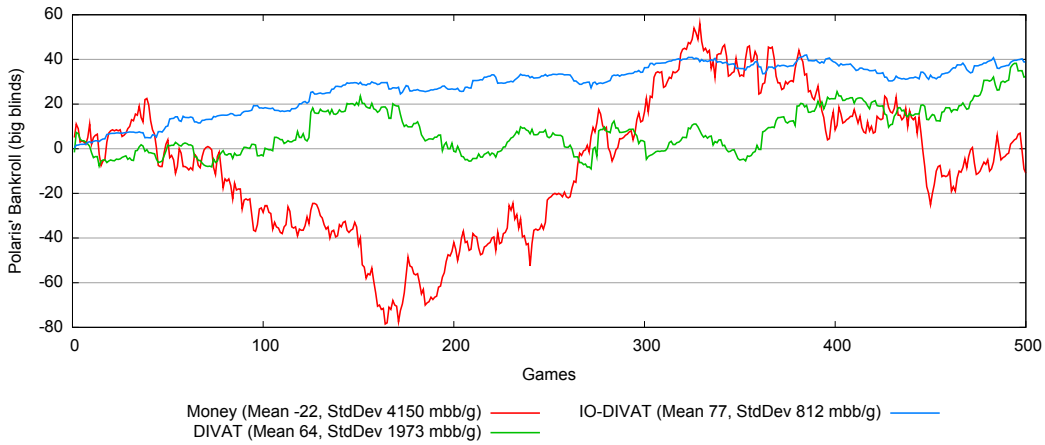


(b) Ali Eslami: Hotel Room

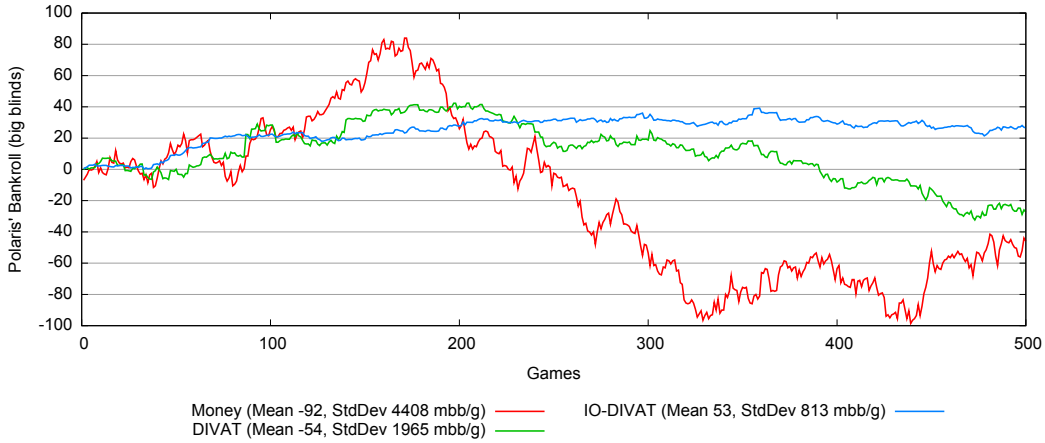


(c) Duplicate

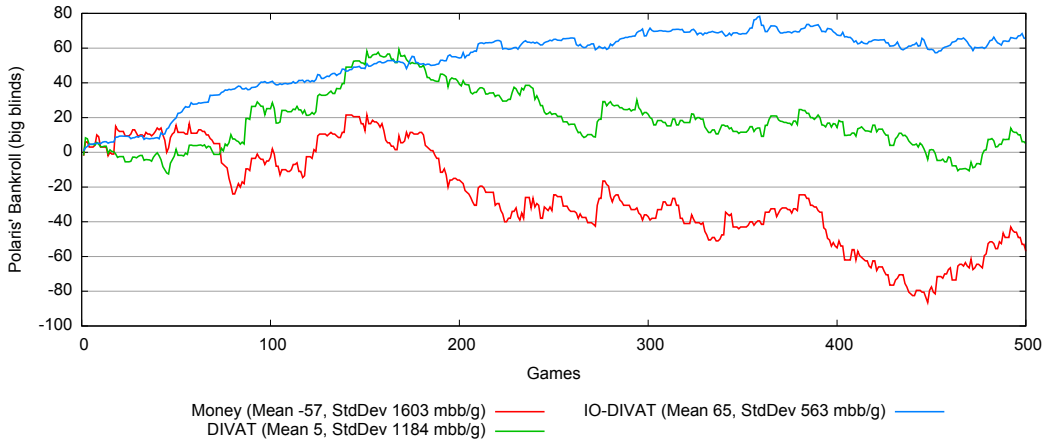
Figure 8.3: 2007 Man-vs-Machine Championship Match 3.



(a) Phil Laak: Hotel Room



(b) Ali Eslami: On Stage



(c) Duplicate

Figure 8.4: 2007 Man-vs-Machine Championship Match 4.

	Match 1	Match 2	Match 3	Match 4	Total
Money	0	1	-1	-1	-1
Significant Money	0	0	0	0	0
DIVAT	0	1	0	0	1
Significant DIVAT	0	0	0	0	0
IO-DIVAT	1	1	1	1	4
Significant IO-DIVAT	1	1	0	1	3

Table 8.1: Summary table for the 2007 First Man-vs-Machine Poker Championship. Each pair of rows shows analysis by a different unbiased postgame analysis technique, with "Money" indicating the raw outcome of each match. 1, 0, or -1 indicate a win, tie or loss for Polaris.

In Table 8.1, we summarize these results in a summary table. Each pair of rows applies one of the three value estimators to each duplicate match outcome, and states whether Polaris won (1) tied (0) or lost (-1) that match. The first row of each pair uses the estimator's raw outcome, while the second row declares a tie unless the winner's margin of victory has 95% confidence in a one-sided significance test.

The Money and DIVAT estimators predict a loss and a win for Polaris respectively, but none of the matches achieve statistical significance through this analysis. The Imaginary Observations technique, however, estimates that Polaris had an advantage in all four matches, and that in three of these matches Polaris' edge was significant according to the one-sided 95% confidence test.

Although Polaris lost the First Man-vs-Machine Poker Championship, we learned a great deal by competing against Phil Laak and Ali Eslami. We are grateful to them for their time and expertise which they openly shared with us. In practical terms, we identified several areas for improvement that guided our research over the next year: specifically, the use of imperfect recall abstractions to better represent the game state, and the importance of aggressive and exploitive strategies and online adaptation to the opponent.[4]

8.2 The Second Man-vs-Machine Poker Championship, 2008

In 2008, after a significant year's progress on Polaris, we were ready for a rematch against human professionals. The Second Man-vs-Machine Poker Championship was held in July in Las Vegas, at the Gaming Life Expo convention held alongside

[4]Our single victory, in Match 2, was scored by an aggressive (non-equilibrium) strategy that played a "tilted" strategy, described in Chapter 2. Laak and Eslami found this to be a much more challenging opponent than the equilibrium stratgy used in both Match 1 and 4, even though the equilibrium strategy used a card abstraction four times larger than the aggressive strategy. This agent was described in detail in my Masters thesis [47, Section 7.3.2].

the World Series of Poker.[5] In this second championship we challenged a group of heads-up limit Texas hold'em specialists, organized by the StoxPoker poker training website. One of our opponents, Matt "Hoss_TBF" Hawrilenko, was widely regarded by other professional poker players to be the world's strongest human player at heads-up limit Texas hold'em. He was joined by Nick "StoxTrader" Grudzien, IJay "doughnutz" Palansky, Kyle "cottonseed" Hendon, Mark Newhouse, Victor Acosta, and Rich McRoberts.

The event consisted of a set of six 500-hand duplicate matches played against pairs of humans from the team. The first two were played online, and the final four were played live in Las Vegas. As in the 2007 match, one human player played in front of an audience in the expo hall, while the other played in a hotel room.

We had made several improvements to Polaris in the year between 2007 and 2008. Most notably, Polaris used the online off-policy partial information Imaginary Observations technique to switch between five component strategies during each match. This technique allowed us to quickly find and use the most appropriate exploitive strategy to use against the opponent's most recent behaviour. Following our earlier colour-based naming convention from the 2007 match, one of the strategies (Pink) was an abstract game Nash equilibrium approximation, while the other four (Orange, Peach, Red and Green) were "tilted" in different ways to create aggressive strategies, as previously described in Figure 2.5. The five component strategies all used an imperfect recall card abstraction that nested public texture and percentile hand strength private buckets. Pink, Orange and Peach had 266,135,752 information sets and were twice as large as the largest Polaris 2007 strategy, while Red and Green had 115,386,552 information sets and were approximately the same size. While seemingly large at the time, these abstractions are tiny compared to those now commonly in use.

For the live matches, one 500-hand match was played per day over four consecutive days. As in 2007, a 25 big blind margin in duplicate score was required to declare a winner. Over two remote matches and four live matches, Polaris won three matches, lost two and tied one, marking the first time that a poker program had ever defeated human professionals in a meaningful poker match. The hand histories and overall money scores were revealed in real time on the competition website [75]. However, as in 2007, the variance reduction provided by the duplicate match format was insufficient for any of the six matches to have a statistically significant outcome.

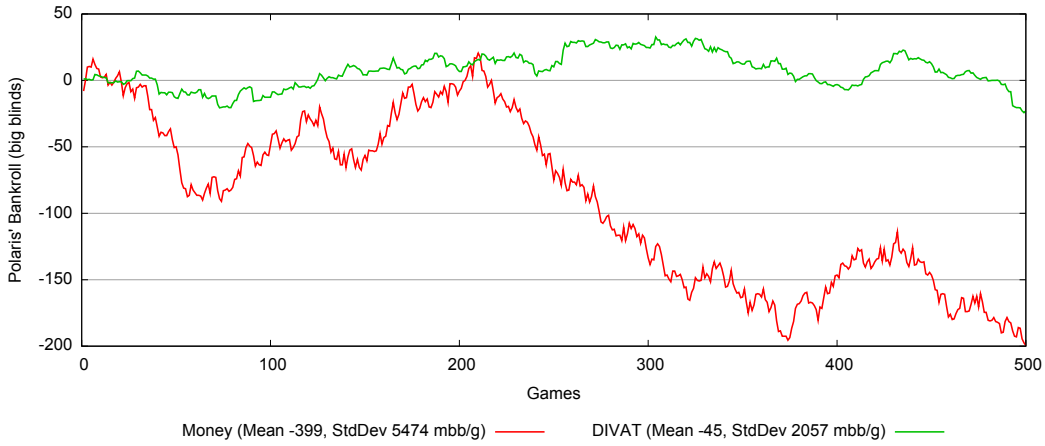
For the first time, we will now present an analysis of these hands using the DIVAT and Imaginary Observations variance reduction techniques.[6] Using these

[5] See [75] for the website used during the event, which includes photos, media links, and a minute-by-minute commentary of each match.

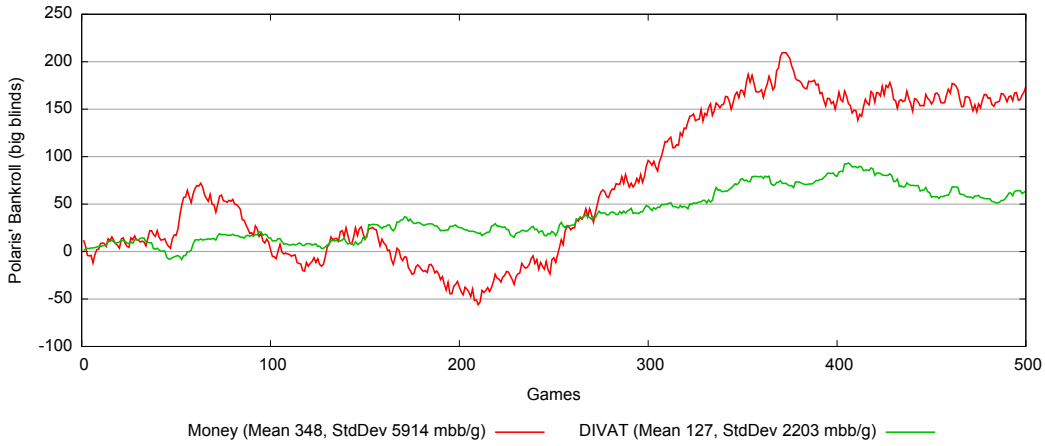
[6] Note that there are two distinct uses of Imaginary Observations being used in this match. During the match, Polaris used online off-policy partial-information Imaginary Observations to choose which strategy to play. This online analysis was biased due to the use of off-policy evaluation, and also because of the use of DIVAT only at showdowns. DIVAT requires all players' card information after a game, and this is not available during a match after either player folds. In the postgame analysis that we will present here, however, our use of DIVAT and Imaginary Observations is unbiased.

techniques, we can now show that Polaris earned its victory in the 2008 Man-vs-Machine match. The results of the two remote matches and four live matches are shown in Figures 8.5c through 8.10c.

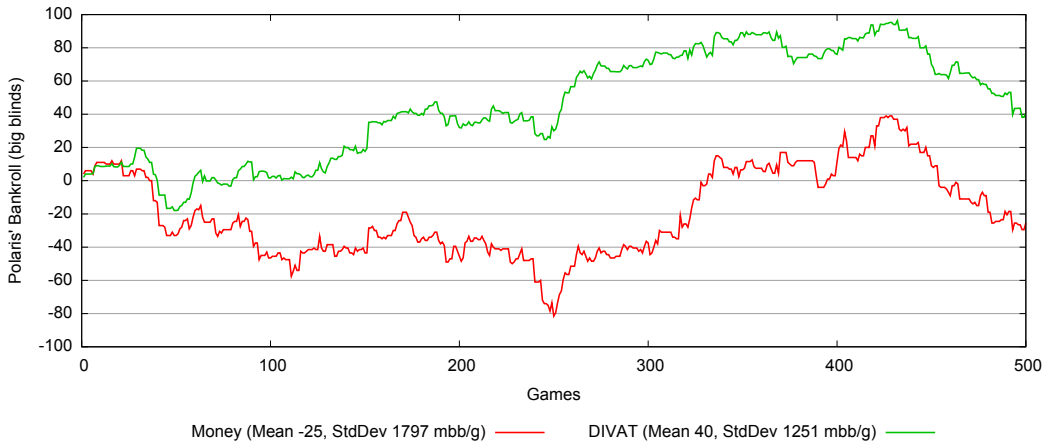
This is possible because for postgame analysis we have the full-information hand histories (revealing all players' cards on each hand), and our analysis is on-policy since we know which strategy Polaris used in each game.



(a) Matt Hawrilenko

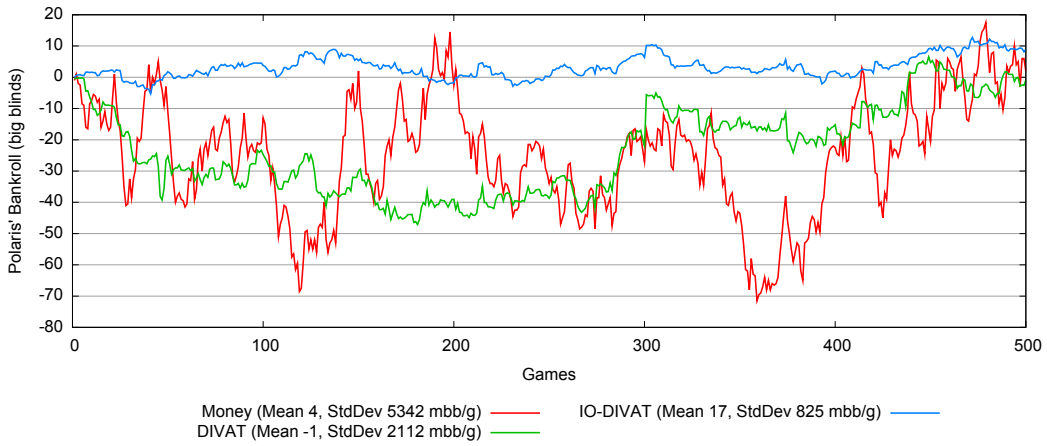


(b) IJay Palansky

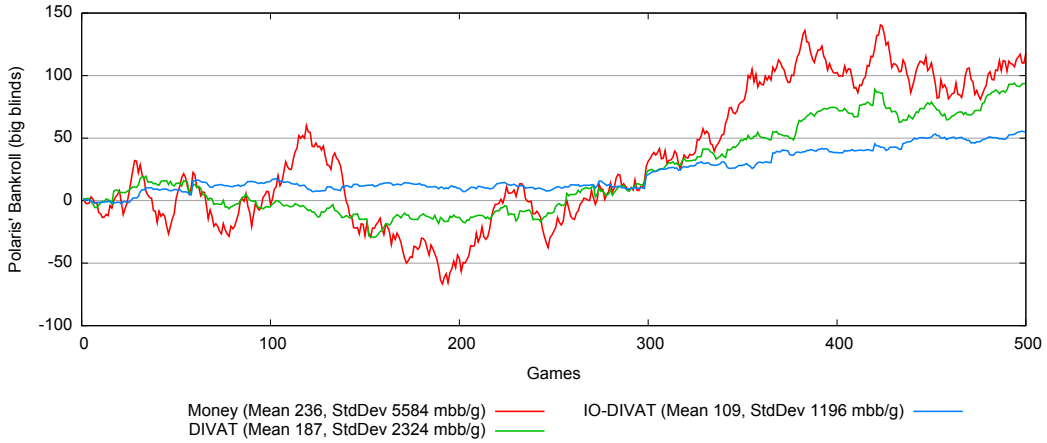


(c) Duplicate

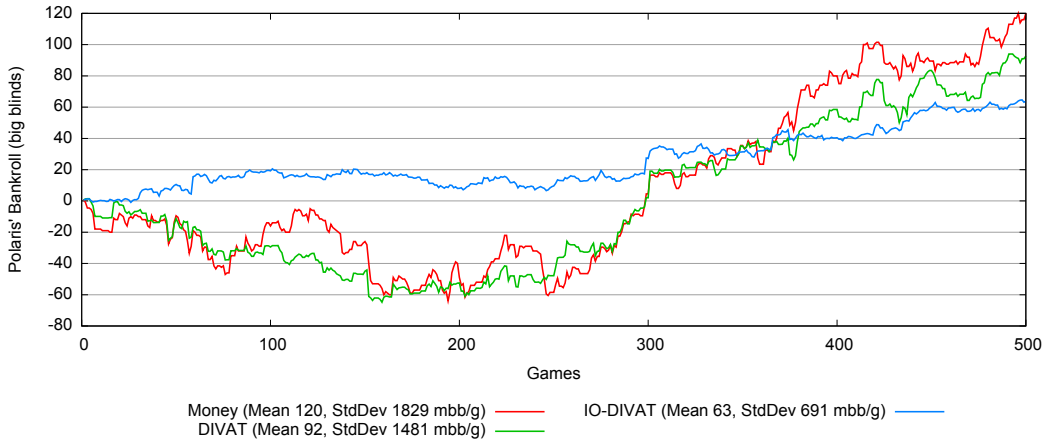
Figure 8.5: 2008 Man-vs-Machine Championship Remote Match 1. Due to a technical error during this match the record of which strategy Polaris chose to use for each hand was lost, and so post-game analysis with Imaginary Observations is not possible.



(a) Nick Grudzien

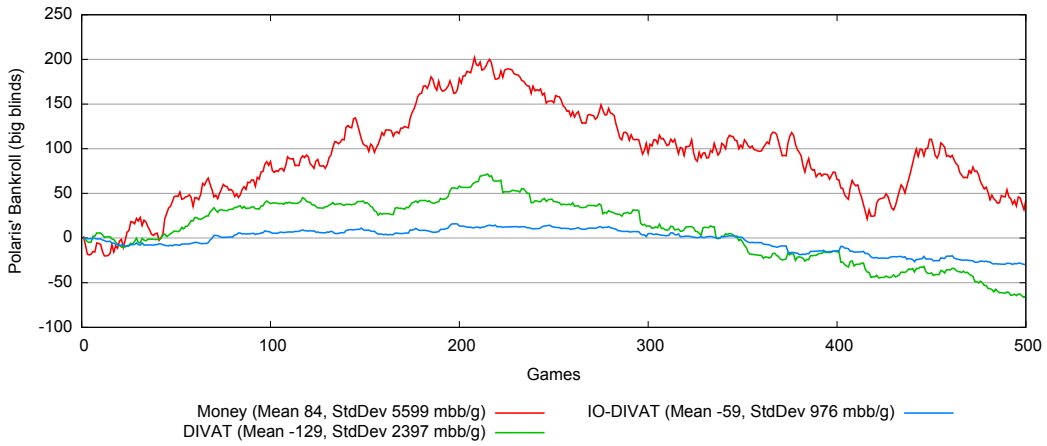


(b) Kyle Hendon

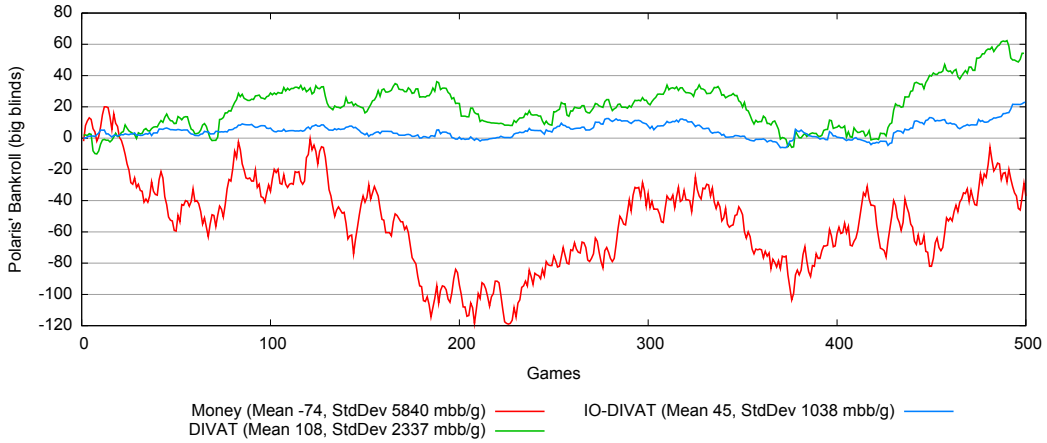


(c) Duplicate

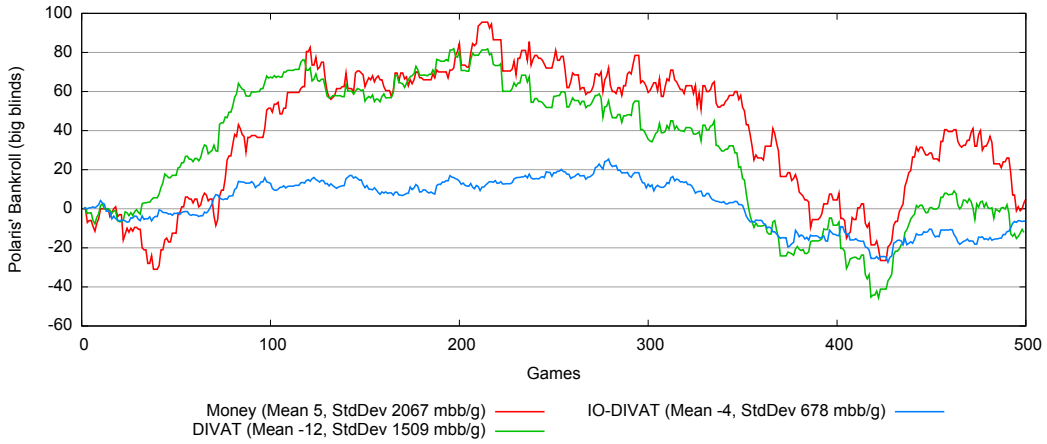
Figure 8.6: 2008 Man-vs-Machine Championship Remote Match 2.



(a) Nick Grudzien: On Stage

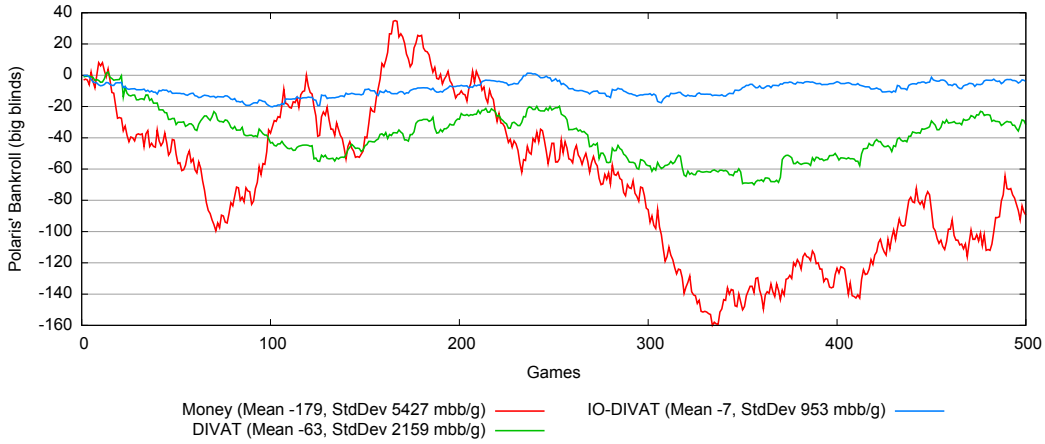


(b) Kyle Hendon: Hotel Room

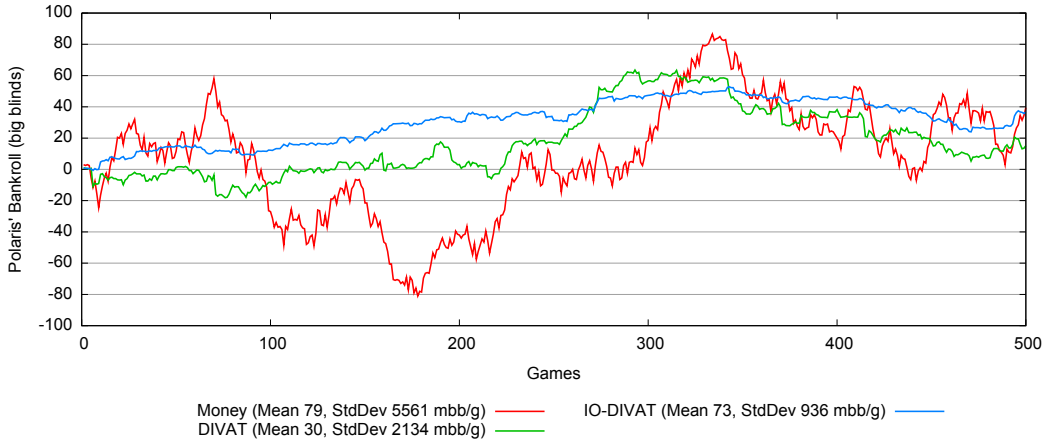


(c) Duplicate

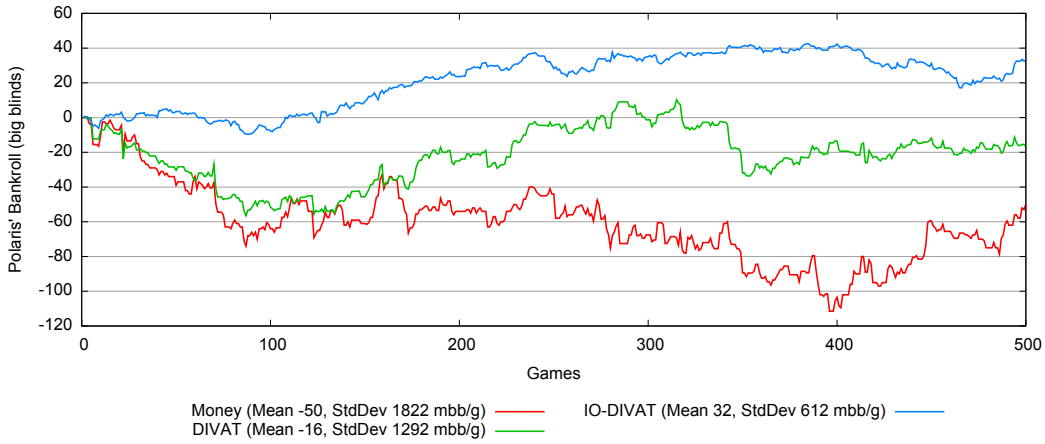
Figure 8.7: 2008 Man-vs-Machine Championship Live Match 1.



(a) Rich McRoberts: On Stage

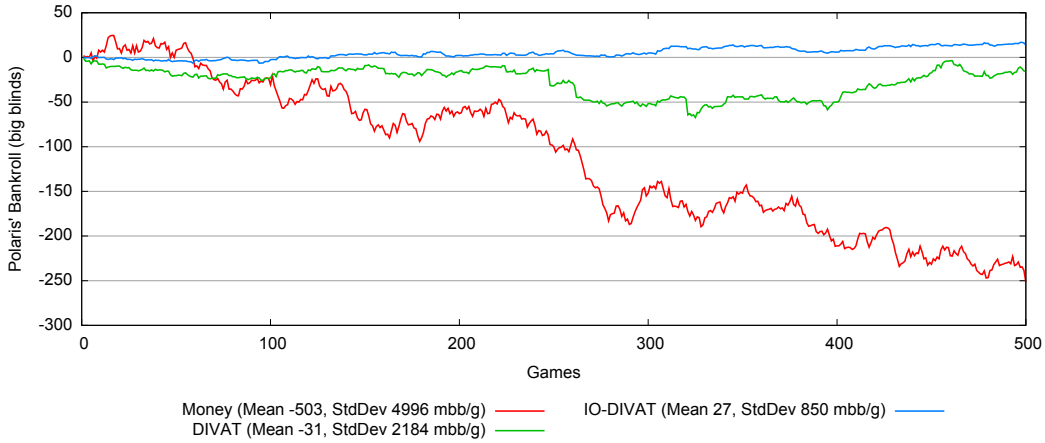


(b) Victor Acosta: Hotel Room

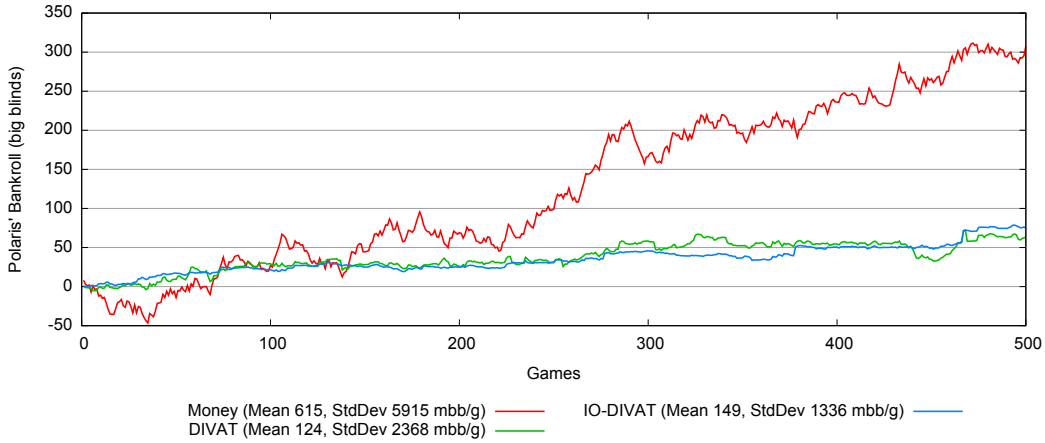


(c) Duplicate

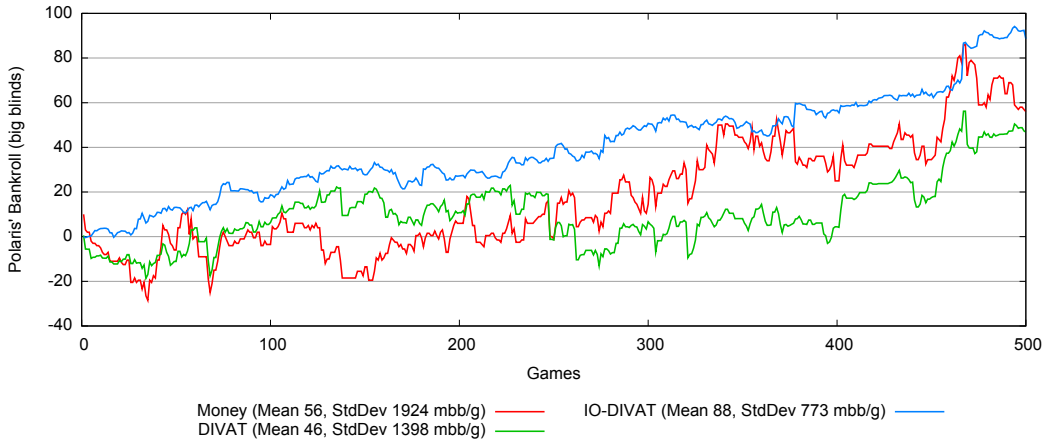
Figure 8.8: 2008 Man-vs-Machine Championship Live Match 2.



(a) Mark Newhouse: On Stage

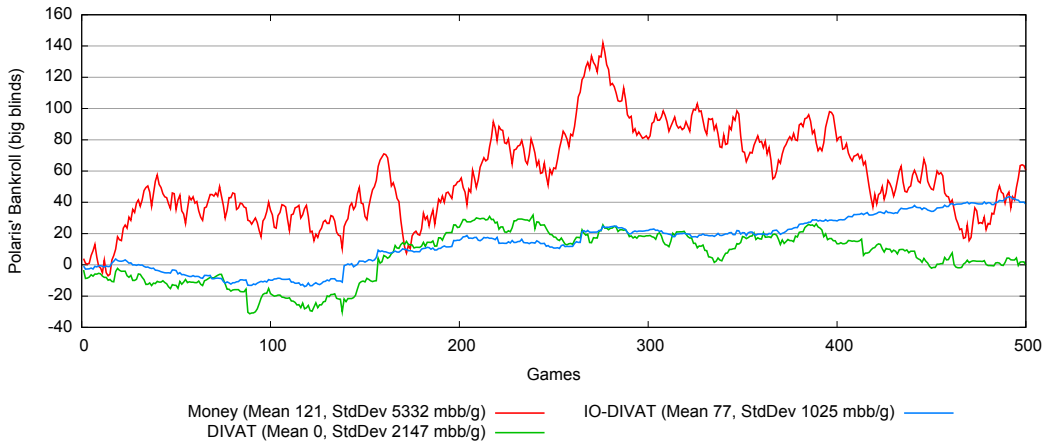


(b) IJay Palansky: Hotel Room

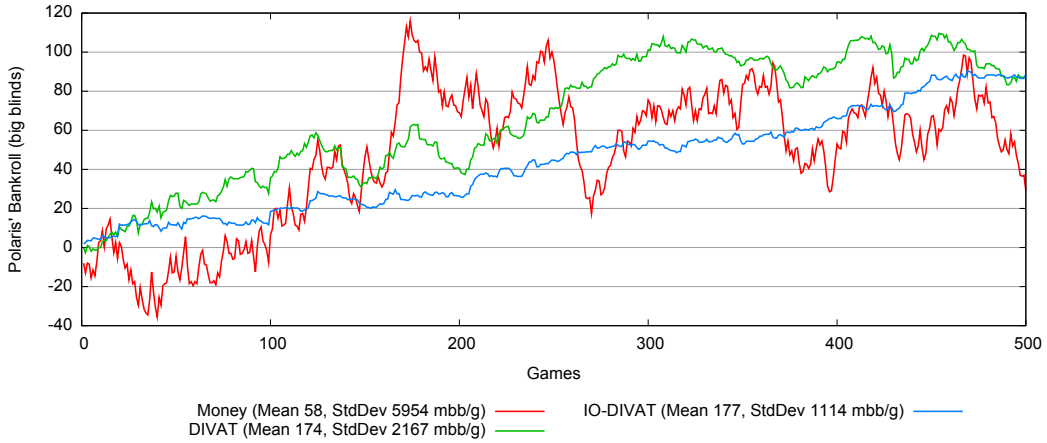


(c) Duplicate

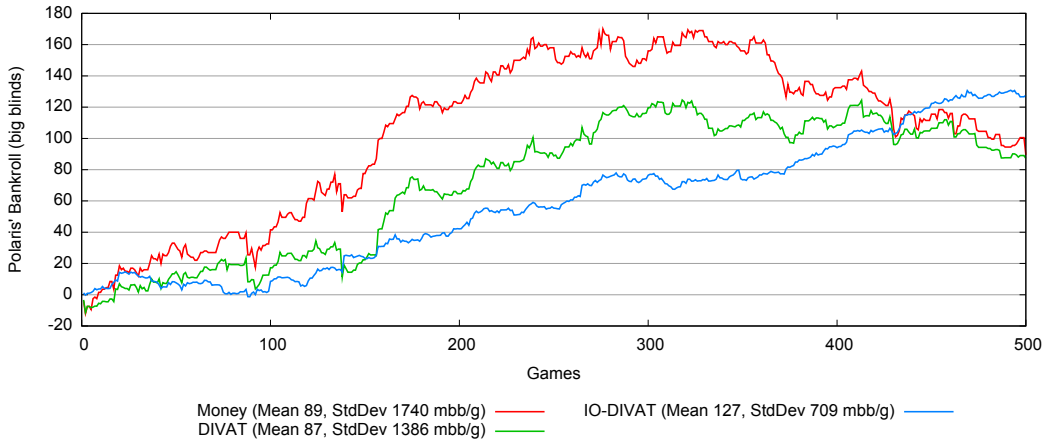
Figure 8.9: 2008 Man-vs-Machine Championship Live Match 3.



(a) Matt Hawrilenko: On Stage



(b) IJay Palansky: Hotel Room



(c) Duplicate

Figure 8.10: 2008 Man-vs-Machine Championship Live Match 4.

	R1	R2	1	2	3	4	Total
Money	-1	1	0	-1	1	1	1
Significant Money	0	0	0	0	0	0	0
DIVAT	1	1	-1	-1	1	1	2
Significant DIVAT	0	0	0	0	0	0	0
IO-DIVAT	–	1	-1	1	1	1	3
Significant IO-DIVAT	–	1	0	0	1	1	3

Table 8.2: Summary table for the 2008 Second Man-vs-Machine Poker Championship. Each pair of rows shows analysis by a different unbiased postgame analysis technique. 1, 0, or -1 indicate a win, tie or loss for Polaris.

In Table 8.2, we summarize our analysis of the six matches. Matches "R1" and "R2" refer to the two remote matches, while matches 1 through 4 refer to the live matches played in Las Vegas. As with our summary of the 2007 event, a 1, 0, or -1 indicates a predicted win, tie, or loss for Polaris. Each pair of rows represents one of our analysis techniques, with the first row declaring a winner if the margin was greater than 25 big blinds. The second "Significant" row declares a winner if their score was greater than 0 with 95% confidence in a one-sided test.

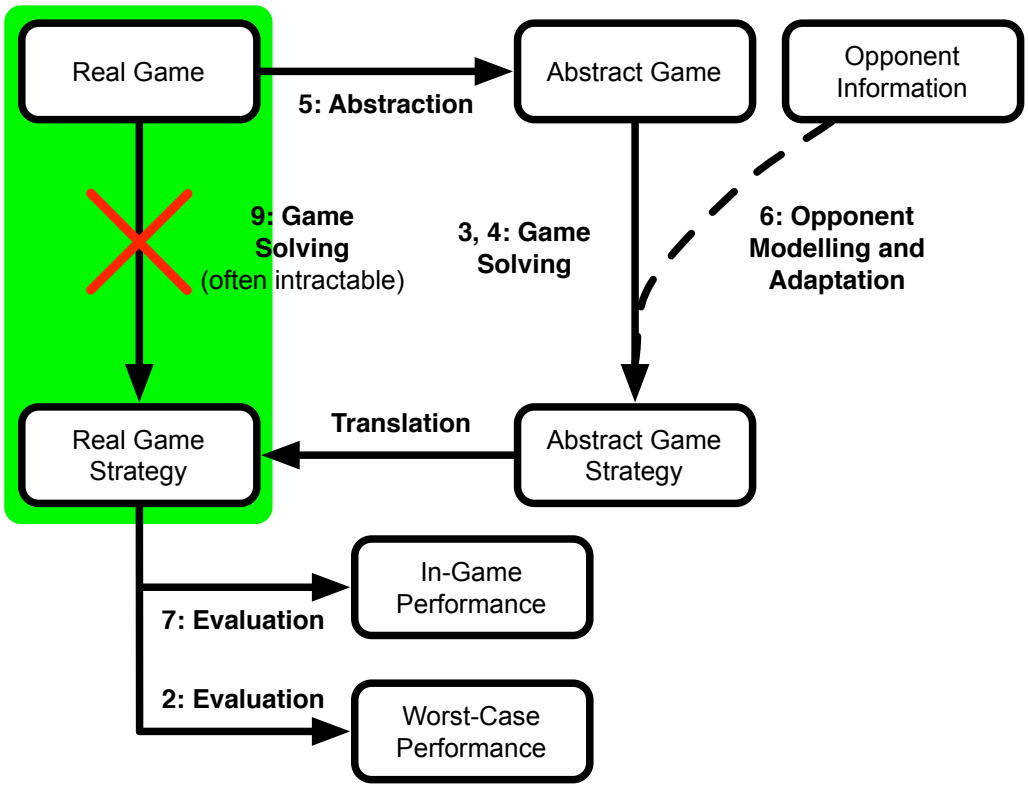
Using the raw Money score used in the competition, Polaris won the event with three wins, two losses and one draw, but none of these results were statistically significant. DIVAT analysis raises Polaris' predicted winnings to four wins and two losses, with none being statistically significant. As noted in the caption of Figure 8.5, due to a technical error, Imaginary Observations cannot be applied to Remote Match 1. Across the other five matches, it predicts four wins and one loss, with three of the wins being statistically significant. This result supports Polaris' victory in the Second Man-vs-Machine Poker Championship.

In the ACPC events after 2008, our agent Hyperborean and other strong ACPC competitors have continued to improve beyond Polaris 2008's strength. While match outcomes are not always transitive, we predict that most of the current ACPC heads-up limit Texas hold'em competitors would defeat Polaris 2008, and would also defeat top human players. Competition became so tight in the ACPC heads-up limit events that the heads-up limit "Instant Runoff" event was dropped in 2014. And now that my colleagues and I have succeeded in solving the game, there will be no need for a third match in this variant of poker. Research leading to further human-computer events will continue in more complicated variants of poker, such as no-limit Texas hold'em.

Chapter 9

Heads-up Limit Hold'em Poker is Solved

Heads-up Limit Hold'em Poker is Solved
Michael Bowling, Neil Burch, Michael Johanson, Oskari Tammelin
Published in Science, 2015 [17]



When we create artificially intelligent agents for a game, there are two significant milestones: the first time that a program defeats the top-ranked human players and surpasses human abilities, and the first time that a program solves the game, such that it will never lose again against any adversary. In the last chapter, we described our 2008 Man-vs-Machine Poker Championship, when Polaris defeated human champion players for the first time to reach the first milestone. In January 2015, our new agent Cepheus[1] reached the second milestone in the game of heads-up limit Texas hold'em by essentially solving the game. With an exploitability of 0.986 mbb/g, its approximation of a Nash equilibrium is so close that even a human lifetime of perfect play – 60 million games – is insufficient to have 95% confidence of defeating Cepheus.

While other human-scale games have been solved previously, such as checkers [85], Connect-Four [2], and awari [82], our solving of heads-up limit Texas hold'em marked the first time that any human-scale imperfect information game had been solved. Furthermore, it is among the first human-scale stochastic games to be solved: the only predecessor is hypergammon, a three-checker version of backgammon. The full game of heads-up limit Texas hold'em is also three orders of magnitude larger than any abstraction of a poker game that has been solved previously. While the accomplishment is exciting within the poker domain, its significance is much greater as it illustrates the power of modern game theoretic algorithms and the scope of problem that can now be efficiently reasoned about.

Our effort to solve heads-up limit Texas hold'em has spanned more than a decade, including the entirety of my Masters and PhD degrees. Our earlier attempts to meet this goal are present in the papers contained in this thesis. Prior to the development of Accelerated Best Response, we believed that if we could solve a large and accurate enough abstraction, we might be able to drive exploitability low enough to solve the game. Unfortunately, even in large abstractions, abstraction pathologies and abstraction overfitting meant that this approach was unlikely to succeed. With the development of CFR-BR, we hoped that solving an abstract game *properly* might work. But even in very large abstractions (see Appendix 11.2), our improvement in exploitability with respect to abstraction size was converging at a rate that suggested that we would only reach 1 mbb/g when the abstract game was as large as the real game. For a third attempt, Burch developed the CFR-D algorithm [21], which safely uses decomposition to solve a game and dramatically reduces the memory required, in exchange for an increased computational cost. This allowed us to avoid using abstraction entirely, but the greatly increased computational cost was infeasible in practice.

Finally, a new variant of CFR called CFR+ provided a solution. CFR+ was developed by Oskari Tammelin [94], an independent programmer from Finland. CFR+ uses the vector-style updates first used by PCS but without any use of sam-

[1] Cepheus is named for the constellation Cepheus, which contains the star Gamma Cephei. While the star Polaris is currently the Earth's northern pole star, around the year 3000 Gamma Cephei will replace it. Thus the star Gamma Cephei is replacing the star Polaris, our poker agent Cepheus is replacing our earlier agent Polaris, and the algorithm CFR+ used to create Cepheus may also replace CFR.

pling techniques, and makes a few subtle changes to the algorithm that result in a greatly improved convergence rate over other CFR variants. Recently, Tammelin, Bowling, Burch and I published a followup paper [95] that establishes the theoretical support for CFR+, and may help to explain this surprisingly fast convergence. This greatly improved convergence rate overcame the computational cost associated with solving a game as large as heads-up limit Texas hold'em. Tammelin also developed a streaming compression algorithm, optimized for poker games, that allowed us to compress CFR+'s internal variables during the computation, reducing the effective memory cost from 523 TB to 11 TB. By distributing the computation across a cluster of 200 identical nodes with 4800 CPU cores total, and making use of local hard disks on each node, the joint team of Bowling, Burch, Tammelin and myself were able to essentially solve the game in just over 900 CPU-years of computation. We unveiled the resulting strategy in detail on our website [16], where visitors can play against Cepheus and inspect its strategy at arbitrary points.

This result largely completes the story of AI for heads-up limit Texas hold'em, and creates new possibilities for solving larger and more complex games. While researchers have long been exploring the larger and more popular poker game of no-limit Texas hold'em with equal vigor, the prospect of being able to solve abstractions one thousand times larger than was possible just months ago has renewed our hopes of defeating human professionals in this domain as well. Our understanding of CFR+ is also still incomplete, which makes it an exciting area for research. We have now proven that it is guaranteed to converge [95], and have demonstrated empirically that it converges far faster than other CFR variants, but have not yet proven that its convergence bound has improved; further, we have only intuition for why its speed has increased. CFR+ also does not currently work well with sampling, which is vital for solving abstract games. If these gaps can be addressed, then CFR+ may overtake CFR as the new state-of-the-art game solving technique.

Author's contributions. Tammelin invented the CFR+ algorithm and streaming compression technique used in the paper. Our implementation of CFR+, designed for high performance clusters, was written and tested by Burch and myself. I was responsible for tuning parameters to find an acceptable three-way tradeoff between solution quality, disk and RAM memory required, and computation time. I also ran the three-month long computation and collected the empirical results. Bowling led our decade-long effort to solve the game and was responsible for composing the manuscript. Burch and I assisted with manuscript editing and formatting.

Heads-up Limit Hold'em Poker is Solved[2]

Michael Bowling (bowling@cs.ualberta.ca)
Neil Burch (burch@ualberta.ca)
Michael Johanson (johanson@ualberta.ca)
Oskari Tammelin (ot@iki.fi)

Abstract:
Poker is a family of games that exhibit imperfect information, where players do not have full knowledge of past events. While many perfect information games have been solved (e.g., Connect-Four and checkers), no nontrivial imperfect information game played competitively by humans has previously been solved. In this paper, we announce that the smallest variant of poker in-play, heads-up limit Texas hold'em, is now essentially weakly solved. Furthermore, this computation formally proves the common wisdom that the dealer in the game holds a significant advantage. This result was enabled by a new algorithm, CFR$^+$, which is capable of solving extensive-form games three orders of magnitude larger than previously possible.

9.1 Introduction

Games have been intertwined with the earliest developments in computation, game theory, and artificial intelligence (AI). At the very conception of computing, Babbage had detailed plans for an "automaton" capable of playing tic-tac-toe and dreamt of his Analytical Engine playing chess [6, Chapter 34]. Both Alan Turing [97] and Claude Shannon [87], on paper and in hardware respectively, developed programs to play chess as validation of early ideas in computation and AI. For over a half-century, games have continued to act as testbeds for new ideas and the resulting successes have marked significant milestones in the progress of AI: e.g., the checkers-playing computer program Chinook becoming the first to win a world championship title against humans [86], Deep Blue defeating Kasparov in chess [25], and Watson defeating Jennings and Rutter on Jeopardy! [29]. However, defeating top human players is not the same as "solving" a game, i.e., computing a game-theoretically optimal solution that is incapable of losing against any opponent in a fair game. Solving games has also served as notable milestones for the advancement of AI, e.g., Connect-Four [2] and checkers [85].

Every nontrivial game played competitively by humans that has been solved to-date is a *perfect information game*.[3] In perfect information games, all players are

[2]The paper presented in this chapter originally appeared in the journal *Science*. Copyright 2015 by the authors. M. Bowling, N. Burch, M. Johanson and O. Tammelin. Heads-up Limit Hold'em Poker is Solved. *Science*, Volume 347 Issue 6218, 145-149, January 2015.

[3]We use the word trivial to describe a game that can be solved without the use of a machine. The one near-exception to this claim is oshi-zumo, but it is not played competitively by humans and is a simultaneous-move game that otherwise has perfect information [23]. Furthermore, almost all nontrivial games played by humans that have been solved to-date also have no chance elements. The one notable exception is hypergammon, a three-checker variant of

informed of everything that has occurred in the game prior to making a decision. Chess, checkers, and backgammon are examples of perfect information games. In *imperfect information games*, players do not always have full knowledge of past events (e.g., cards dealt to other players in bridge and poker, or a seller's knowledge of the value of an item in an auction). These games are more challenging, with theory, computational algorithms, and instances of solved games lagging behind results in the perfect information setting.[4] And, while perfect information may be a common property of parlour games, it is far less common in real-world decision making settings. In a conversation recounted by Bronowski, John von Neumann, the founder of modern game theory, made the same observation, "Real life is not like that. 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." [20].

Von Neumann's statement hints at the quintessential game of imperfect information: the game of poker. Poker involves each player being dealt private cards, with players taking structured turns making bets on having the strongest hand (possibly bluffing), calling opponent bets, or folding to give up the hand. Poker played an important role in the early developments of the field of game theory. Borel [15] and von Neumann's [99; 100] foundational works were motivated by developing a mathematical rationale for bluffing in poker, and small synthetic poker games[5] were commonplace in many early papers [15; 100; 60; 72]. Poker is also arguably the most popular card game in the world with over 150 million players worldwide [1]. The most popular variant of poker today is Texas hold'em. When it is played with just two-players (heads-up) and with fixed bet-sizes and number of raises (limit), it is called heads-up limit hold'em or HULHE. HULHE was popularized by a series of high-stakes games chronicled in the book *The Professor, the Banker, and the Suicide King* [27]. It is also the smallest variant of poker played competitively by humans. HULHE has 3.16×10^{17} possible states the game can reach making it larger than Connect Four and smaller than checkers. However, as an imperfect information game, many of these states cannot be distinguished by the acting player as they involve information about unseen past events (i.e., private cards dealt to the opponent). As a result, the game has 3.19×10^{14} decision points where a player is required to make a decision.

While smaller than checkers, the imperfect information nature of HULHE makes it a far more challenging game for computers to play or solve. It was 17 years after

backgammon invented by Hugh Sconyers in 1993 which he then strongly solved, i.e., the game-theoretic value is known for all board positions. It has seen play in human competitions. See http://www.bkgm.com/variants/HyperBackgammon.html (accessed July 4, 2014).

[4]For example, Zermelo proved the solvability of finite, two-player, zero-sum, perfect information games in 1913 [106], while von Neuman's more general minimax theorem appeared in 1928 [99]. Minimax and alpha-beta pruning, the fundamental computational algorithm for perfect information games, was developed in the 1950s, while Koller and Megiddo's first polynomial-time technique for imperfect information games was introduced in 1992 [56].

[5]We use the word synthetic to describe a game that was invented for the purpose of being studied or solved rather than played by humans. A synthetic game may be trivial, such as Kuhn poker [60], or nontrivial such as Rhode Island hold'em [89]

Chinook won its first game against world champion Marion Tinsley in checkers that the computer program Polaris won the first meaningful match against professional poker players [78]. While Schaeffer et al. solved checkers in 2007 [85], heads-up limit Texas hold'em poker, until now, was unsolved. This slow progress is not for lack of effort. Poker has been a challenge problem for artificial intelligence, operations research, and psychology with work going back over 40 years [13]. 17 years ago, Koller and Pfeffer [59] declared, "we are nowhere close to being able to solve huge games such as full-scale poker, and it is unlikely that we will ever be able to do so." The focus on HULHE as one example of "full-scale poker" began in earnest over ten years ago [11], and became the focus of dozens of research groups and hobbyists after 2006 when it became the inaugural event in the Annual Computer Poker Competition [65], held in conjunction with the main conference of the Association for the Advancement of Artificial Intelligence (AAAI). This paper is the culmination of this sustained research effort toward solving a "full-scale" poker game.

Allis [3] gives three different definitions for solving a game. A game is said to be *ultra-weakly solved* if for the initial position(s), the game-theoretic value has been determined; *weakly solved* if for the initial position(s), a strategy has been determined to obtain at least the game-theoretic value, for both players, under reasonable resources; and *strongly solved* if for all legal positions, a strategy has been determined to obtain the game-theoretic value of the position, for both players, under reasonable resources. In an imperfect information game, where the game-theoretic value of a position beyond the initial position is not unique, Allis's notion of "strongly solved" is not well-defined. Furthermore, imperfect information games, due to stochasticity in the players' strategies or the game itself, typically have game-theoretic values that are real-valued rather than discretely valued (such as "win", "loss", and "draw" in chess and checkers), and only achieved in expectation over many playings of the game. As a result, game-theoretic values are often approximated, and so an additional consideration in solving a game is the degree of approximation in a solution. A natural level of approximation under which a game is *essentially weakly solved* is if a human lifetime of play is not sufficient to establish with statistical significance that the strategy is not an exact solution.

In this paper, we announce that heads-up limit Texas hold'em poker is essentially weakly solved. Furthermore, we bound the game-theoretic value of the game, proving that the game is a winning game for the dealer.

9.2 Solving Imperfect Information Games

The classical representation for an imperfect information setting is the *extensive-form game*. Here the word "game" refers to a formal model of interaction between self-interested agents and applies to both recreational games and serious endeavours such as auctions, negotiation, and security. See Figure 9.1 for a graphical depiction of a portion of a simple poker game in extensive-form. The core of an extensive-form game is a *game tree* specifying branches of possible events, namely player

actions or chance outcomes. The branches of the tree split at *game states* and each is associated with one of the players (or chance) who is responsible for determining the result of that event. The leaves of the tree signify the end of the game, and have an associated utility for each player. The states associated with a player are partitioned into *information sets*, which are sets of states which the acting player cannot distinguish between (e.g., corresponding to states where the opponent was dealt different private cards). The branches from states within an information set are the player's available *actions*. A *strategy* for a player specifies for each information set a probability distribution over the available actions. If the game has exactly two players and the utilities at every leaf sum to zero, the game is called *zero-sum*.

The classical solution concept for games is a *Nash equilibrium*, a strategy for each player such that no player can increase their expected utility by unilaterally choosing a different strategy. All finite extensive-form games have at least one Nash equilibrium. In zero-sum games, all equilibria have the same expected utilities for the players, and this value is called the *game-theoretic value of the game*. An ϵ*-Nash equilibrium* is a strategy for each player where no player can increase their utility by more than ϵ by choosing a different strategy. By Allis's categories, a zero-sum game is ultra-weakly solved if its game-theoretic value is computed, and weakly solved if a Nash equilibrium strategy is computed. We call a game essentially weakly solved if an ϵ-Nash equilibrium is computed for a sufficiently small ϵ to be statistically indistinguishable from zero in a human lifetime of played games. For perfect information games, solving typically involves a (partial) traversal of the game tree. However, the same techniques cannot apply to imperfect information settings. We briefly review the advances in solving imperfect information games, benchmarking the algorithms by their progress in solving increasingly larger synthetic poker games as summarized in Figure 9.2.

Normal-Form Linear Programming The earliest method for solving extensive-form games involved converting it into a *normal-form game*, represented as a matrix of values for every pair of possible deterministic strategies in the original extensive-form game, and then solving it with a linear program (LP). Unfortunately, the number of possible deterministic strategies is exponential in the number information sets of the game. So, while LPs can handle normal-form games with many thousands of strategies, even just a few dozen decision points makes this method impractical. Kuhn poker, a poker game with 3 cards, one betting round, and a one bet maximum having a total of 12 information sets (see Figure 9.1), can be solved with this approach. But even Leduc hold'em [90], with 6 cards, two betting rounds, and a two bet maximum having a total of only 288 information sets, is intractable having over 10^{86} possible deterministic strategies.

Sequence-Form Linear Programming Romanovskii [81] and later Koller et al. [56; 57] established the modern era of solving imperfect information games, introducing the sequence-form representation of a strategy. With this simple change of variables, they showed that the extensive-form game could be solved directly as

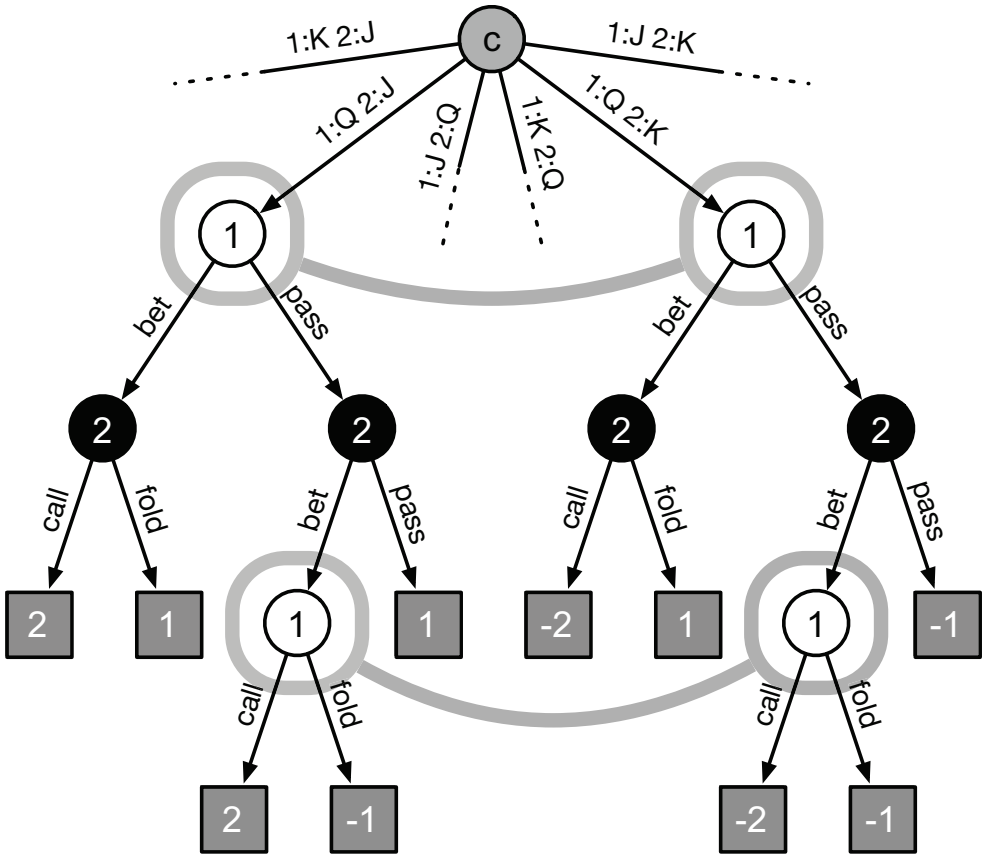


Figure 9.1: Portion of the extensive-form game representation of three-card Kuhn poker [60] where player 1 is dealt a queen (Q) and the opponent is given either the Jack (J) or King (K). Game states are circles labeled by the player acting at each state ("c" refers to chance, which randomly chooses the initial deal). The arrows show the events the acting player can choose from, labeled with their in-game meaning. The leaves are square vertices labeled with the associated utility for player 1 (player 2's utility is the negation of player 1's). The states connected by thick gray lines are part of the same information set, i.e., player 1 cannot distinguish between the states in each pair since they represent a different unobserved card being dealt to the opponent. Player 2's states are also in information sets, containing other states not pictured in this diagram.

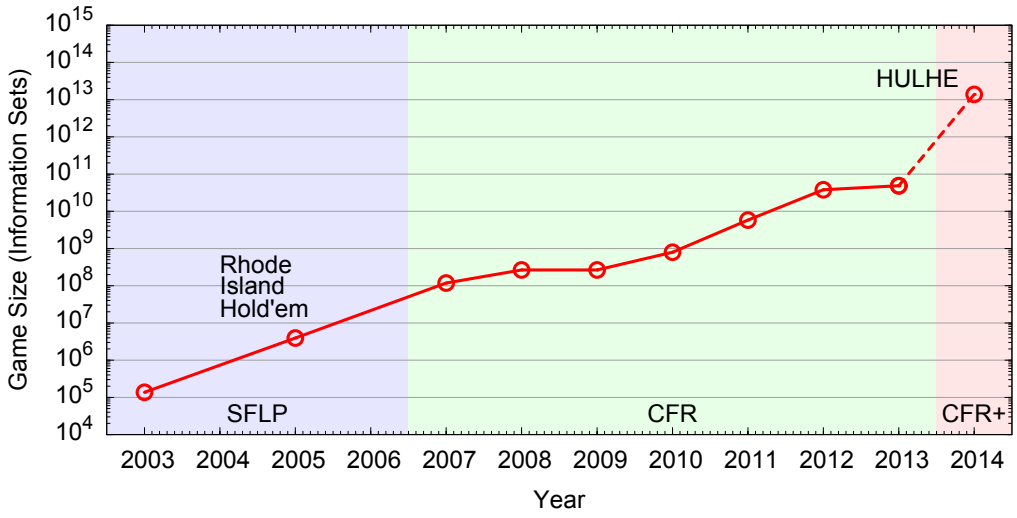


Figure 9.2: Increasing sizes of imperfect information games solved over time. The shaded regions refer to the technique used to achieve the result with references in the main text. CFR$^+$ is the algorithm used in this work and the dashed line shows the result established in this paper.

an LP, without the need for an exponential conversion to normal-form. Sequence-form linear program (SFLP) was the first algorithm to solve imperfect information extensive-form games with computation time that grows as a polynomial of the size of the game representation. In 2003, Billings et al. [11] applied this technique to poker, solving a set of simplifications of HULHE to build the first competitive poker-playing program. In 2005, Gilpin and Sandholm [37] used the approach along with an automated technique for finding game symmetries to solve Rhode Island Hold'em [89], a synthetic poker game with 3.94×10^6 information sets after symmetries are removed.

Counterfactual Regret Minimization In 2006, the Annual Computer Poker Competition was started [65]. The competition drove significant advancements in solving larger and larger games, with multiple techniques and refinements being proposed in the years that followed [84; 83]. One of the techniques to emerge, and currently the most widely adopted in the competition, is counterfactual regret minimization (CFR).[6]. CFR is an iterative method for approximating a Nash equilibrium of an extensive-form game through the process of repeated self-play between two

[6]Another notable algorithm to emerge from the Annual Computer Poker Competition is an application of Nesterov's excessive gap technique [73] to solving extensive form games [34] The technique has some desirable properties, including better asymptotic time complexity that what is known for CFR. However, it has not seen widespread use among competition participants due to its lack of flexibility in incorporating sampling schemes and its inability to be used with powerful (but unsound) abstractions that employ imperfect recall. Recently, Waugh and Bagnell [102] have shown that CFR and the excessive gap technique are more alike than different, suggesting that the individual advantages of each approach may be attainable in the other.

regret-minimizing algorithms [110]. *Regret* is the loss in utility an algorithm suffers for not having selected the single best deterministic strategy, which can only be known in hindsight. A *regret-minimizing* algorithm is one that guarantees its regret grows sub-linearly over time, and so eventually achieves the same utility as the best deterministic strategy. The key insight of CFR is that instead of storing and minimizing regret for the exponential number of deterministic strategies, CFR stores and minimizes a modified regret for each information set and subsequent action, which can be used to form an upper bound on the regret for any deterministic strategy. An approximate Nash equilibrium is retrieved by averaging each player's strategies over all of the iterations, and the approximation improves as the number of iterations increases. The memory needed for the algorithm is linear in the number of information sets, rather than quadratic, which is the case for efficient LP methods [55]. Since solving large games is usually memory-bound, CFR has resulted in as dramatic an increase in the size of solved games as Koller et al.'s advance. Since its introduction in 2007, CFR has been used to solve increasingly complex simplifications of HULHE, reaching as many as 3.8×10^{10} information sets in 2012 [45].

9.3 Solving Heads-Up Limit Hold'em

The full game of HULHE has 3.19×10^{14} information sets. Even after removing game symmetries it has 1.38×10^{13}, i.e., three orders of magnitude larger than previously solved games. There are two challenges for established CFR variants to handle games at this scale: memory and computation. During computation CFR must store the resulting solution and the accumulated regret values for each information set. Even with single-precision (4 byte) floating point numbers, this requires 262 TB of storage. Furthermore, past experience has shown that a three order of magnitude increase in the number of information sets requires at least three orders of magnitude more computation. In order to tackle these two challenges we employ two ideas recently proposed by Tammelin, a co-author of this paper [93].

To address the memory challenge we store the approximate solution strategy and accumulated regrets using compression. For the solution and regrets we use fixed-point arithmetic by first multiplying all values by a scaling factor and truncating them to integers. The resulting integers are then ordered to maximize compression efficiency, with compression ratios around 13-to-1. Overall, we require under 11 TB of storage during the computation, which is distributed across a cluster of computation nodes. This amount is infeasible to store in main memory, and so we store the compressed strategy and regret values on each node's local disk. Each node is responsible for a set of *subgames*, i.e., portions of the game tree partitioned based on publicly observed actions and cards so that each information set is associated with one subgame. The regrets and strategy for a subgame are loaded from disk, updated, and saved back to disk, using a streaming compression technique that decompresses and recompresses portions of the subgame as needed. By making the subgames large enough, the update-time dominates the total time to pro-

cess a subgame. With disk pre-caching, the inefficiency incurred by disk storage is approximately 5% of the total time.

To address the computation challenge we use a variant of CFR called CFR$^+$ [93]. CFR implementations typically sample only portions of the game tree to update on each iteration. They also employ regret-matching at each information set, which maintains regrets for each action and chooses among actions with positive regret with probability proportional to that regret. Instead, CFR$^+$ does exhaustive iterations over the entire game tree, and uses regret-matching$^+$, a variant of regret-matching where regrets are constrained to be non-negative. Actions that have appeared poor (with less than zero regret for not having been played) will be chosen again immediately after proving useful (rather than waiting many iterations for the regret to become positive). Finally, in contrast with CFR, we have observed empirically that the exploitability of the players' strategies during the computation regularly converges to zero. Therefore, we skip the step of computing and storing the average strategy, instead using the players' current strategies as the CFR$^+$ solution. We have empirically observed CFR$^+$ to require considerably less computation than state-of-the-art sampling CFR [50], while also being highly suitable for massive parallelization.

Like CFR, CFR$^+$ is an iterative algorithm that computes successive approximations to a Nash equilibrium solution. The quality of the approximation can be measured by its *exploitability*: the amount less than the game value that the strategy achieves against the worst-case opponent strategy in expectation. Computing the exploitability of a strategy involves computing this worst-case value, traditionally requiring a traversal of the entire game tree. This was long thought to be intractable for games the size of HULHE. Recently it was shown that this calculation could be dramatically accelerated by exploiting the imperfect information structure of the game and regularities in the utilities [53]. This is the technique we use to confirm the approximation quality of our resulting strategy. The technique and implementation has been verified on small games and against independent calculations of the exploitability of simple strategies in HULHE.

A strategy can be exploitable in expectation and yet, due to chance elements in the game and randomization in the strategy, its worst-case opponent still isn't guaranteed to be winning after any finite number of hands. We define a game to be *essentially solved* if a lifetime of play is unable to statistically differentiate it from being solved at 95% confidence. Imagine someone playing 200 hands of poker an hour for 12 hours a day without missing a day for 70 years. Furthermore imagine them employing the worst-case, maximally exploitive, opponent strategy, and never making a mistake. Their total winnings, as a sum of many millions of independent outcomes, would be normally distributed. Hence, the observed winnings in this lifetime of poker would be 1.64 standard deviations or more below its expected value (i.e., the strategy's exploitability) at least 1 time out of 20. Using the standard deviation of a single hand of HULHE, which has been reported to be around 5 bb/g (big-blinds per game, where the big-blind is the unit of stakes in HULHE) [19], we arrive at a threshold of $1.64 * 5 / \sqrt{200 * 12 * 365 * 70} \approx 0.00105$. So, an approximate solution with an exploitability under 1mbb/g (milli-big-blinds

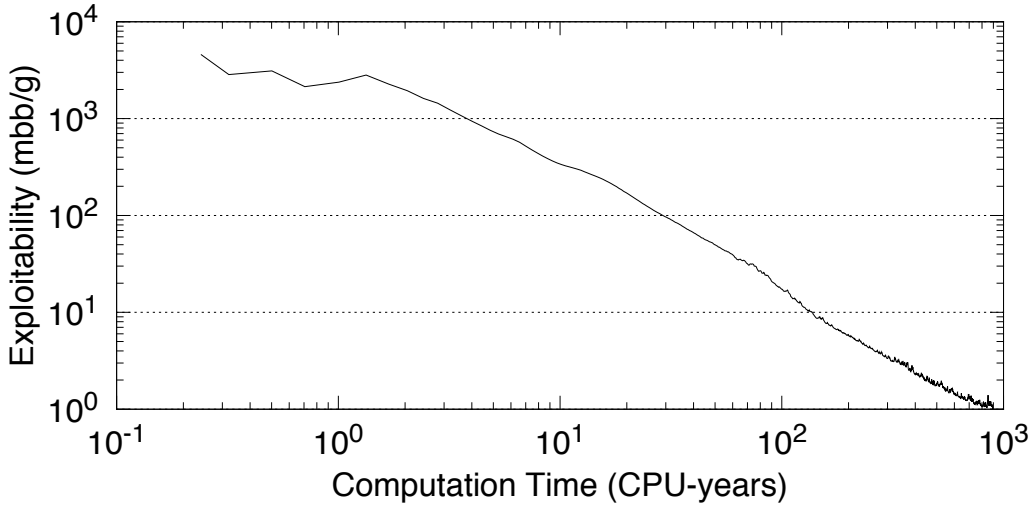


Figure 9.3: Exploitability of the approximate solution with increasing computation.

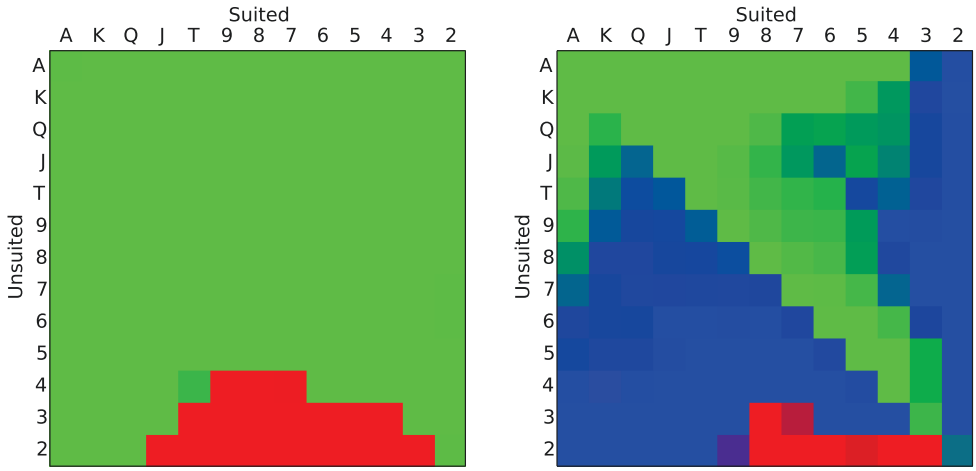
per game) cannot be distinguished with high confidence from an exact solution, and indeed has a 1-in-20 chance of winning against its worst-case adversary even after a human lifetime of games. Hence, 1mbb/g is the threshold for declaring HULHE essentially solved.

9.4 The Solution

Our CFR$^+$ implementation was executed on a cluster of 200 computation nodes each with 24 2.1 GHz AMD cores, 32GB of RAM, and a 1 TB local disk. We divided the game into 110,565 subgames (partitioned based on preflop betting, flop cards, and flop betting). The subgames were split among 199 worker nodes, with one parent node responsible for the initial portion of the game-tree. The worker nodes performed their updates in parallel, passing values back to the parent node for it to perform its update, taking 61 minutes on average to complete one iteration. The computation was then run for 1,579 iterations, taking 68.5 days, and using a total of 900 core years of computation[7] and 10.9 TB of disk space, including filesystem overhead from the large number of files.

Figure 9.3 shows the exploitability of the computed strategy with increasing computation. The strategy reaches an exploitability of 0.986 mbb/g, making HULHE essentially weakly solved. Using the separate exploitability values for each position (as the dealer and non-dealer) we get exact bounds on the game-theoretic value of the game: between 87.7 and 89.7 mbb/g for the dealer, proving the common wisdom that the dealer holds a significant advantage in HULHE.

[7]The total time and number of core years is larger than was strictly necessary as it includes computation of an average strategy that was later measured to be more exploitable than the current strategy and so discarded. The total space noted, on the other hand, is without storing the average strategy.



(a) first action as the dealer

(b) first action as the non-dealer after a dealer raise

Figure 9.4: Action probabilities in the solution strategy for two early decisions. Each cell represents one of the possible 169 hands (i.e., two private cards) with the upper diagonal consisting of cards with the same suit and the lower diagonal consisting of cards of different suits. The color of the cell represents the action taken: red for fold, blue for call, and green for raise, with mixtures of colors representing a stochastic decision.

The final strategy, as a close approximation to a Nash equilibrium, can also answer some fundamental and long-debated questions about game-theoretically optimal play in HULHE. Figure 9.4 gives a glimpse of the final strategy in two early decisions of the game. Human players have disagreed about whether it may be desirable to "limp", i.e., call as the very first action rather than raise, with certain hands. Conventional wisdom is that limping forgoes the opportunity to provoke an immediate fold by the opponent, and so raising is preferred. Our solution emphatically agrees (see the absence of blue in Figure 9.4a). The strategy limps just 0.06% of the time and with no hand more than 0.5%. In other situations, the strategy gives insights beyond conventional wisdom, indicating areas where humans might improve. The strategy rarely "caps", i.e., makes the final allowed raise, in the first round as the dealer, whereas some strong human players cap the betting with a wide range of hands. Even when holding the strongest hand, a pair of aces, the strategy caps the betting less than 0.01%, and the hand most likely to cap is a pair of twos, with probability 0.06%. Perhaps more importantly, the strategy chooses to play, i.e., not fold, a broader range of hands as the non-dealer than most human players (see the relatively small amount of red in Figure 9.4b). It is also much more likely to re-raise when holding a low-rank pair (such as threes or fours).[8]

While these observations are only for one example of game-theoretically optimal play (different Nash equilibria may play differently), they both confirm as well

[8]These insights were the result of discussions with Mr. Bryce Paradis, previously a professional poker player who specialized in HULHE.

as contradict current human beliefs about equilibria play, and illustrate that humans can learn considerably from such large-scale game-theoretic reasoning.

9.5 Conclusion

In this paper, we announced that heads-up limit Texas hold'em poker is essentially weakly solved. This is the first nontrivial imperfect information game played competitively by humans to be solved. Even still, the reader may ask what is the ultimate significance of solving poker? The breakthroughs behind this result are general algorithmic advances that make game-theoretic reasoning in large-scale models of any sort more tractable. And, while seemingly playful, game theory has always been envisioned to have serious implications, e.g., its early impact on cold war politics [69]. More recently, there has been a surge in game-theoretic applications involving security, including systems being deployed for airport checkpoints, air marshall scheduling, and coast guard patrolling [92]. CFR algorithms, based on those described in this paper, have been used for robust decision-making in settings where there is no apparent adversary, with potential application to medical decision support[26]. With real-life decision-making settings almost always involving uncertainty and missing information, algorithmic advances, such as those needed to solve poker, are needed to drive future applications. However, we also echo a response attributed to Alan Turing in defense of his own work in games, "It would be disingenuous of us to disguise the fact that the principal motive which prompted the work was the sheer fun of the thing."[67]

Acknowledgments

The author order is alphabetical reflecting equal contribution by the authors. The idea of CFR$^+$ and compressing the regrets and strategy originated with Oskari Tammelin [94]. This research was supported by Natural Sciences and Engineering Research Council (NSERC), Alberta Innovates Technology Futures (AITF) through the Alberta Innovates Centre for Machine Learning (AICML), and was only possible due to computing resources provided by Compute Canada and Calcul Québec. The authors would like to thank all of the current and past members of the University of Alberta Computer Poker Research Group (CPRG), where the idea to solve heads-up limit Texas hold'em was first discussed; Dr. Jonathan Schaeffer, Dr. Robert Holte, Dr. Duane Szafron, and Dr. Alex Brown for comments on early drafts of this article; and Bryce Paradis for insights into the conventional wisdom of top human poker players.

Supplementary Materials

The implementation of CFR+ that we used to solve HULHE is available online under the open-source BSD license: `http://webdocs.cs.ualberta.ca/~games/poker/cfr_plus.html`

Supplementary material for this paper, providing background information on poker and CFR, is available online [18].

Chapter 10

Conclusion

> "The method of von Neumann and Morgenstern has become the archetype of later applications of game theory. One takes an economic problem, formulates it as a game, finds the game-theoretic solution, then translates the solution back into economic terms."
> – R.J. Aumann [5]

In this thesis, we explored the Abstraction-Solving-Translation procedure as a means of creating strong computer agents for human-scale domains. The techniques we have presented are general, and have advanced the field of computational game theory by reaching two milestones: the first victory of a computer agent over human professionals in a meaningful poker match, and solving the first human-scale imperfect information game. In this final chapter we will summarize the contributions presented in this thesis, note two recent developments in the field, describe promising areas for future work, and conclude with some final remarks.

10.1 Contributions

This thesis presented seven of my research papers as contributions, which span the steps of the Abstraction-Solving-Translation procedure:

- **Accelerating Best Response Calculation in Large Extensive Games (IJCAI 2011)**, [53]. In Chapter 2 we explored the task of evaluating strategies by their exploitability, or distance to a Nash equilibrium. While this computation was previously thought to be intractable for large imperfect information domains, our **Accelerated Best Response** algorithm made the computation possible and provided our first insight in the poker domain. Our experiments showed that the existing strong strategies were still quite exploitable, and revealed the overfitting effect that results from solving abstract games.
- **Efficient Nash Equilibrium Approximation through Monte Carlo Counterfactual Regret Minimization (AAMAS 2012)**, [50]. In Chapter 3, we explored the CFR algorithm for solving large imperfect information games.

Public Chance Sampling CFR improved on the state-of-the-art CFR algorithm by taking advantage of the imperfect information structure of the game, resulting in faster convergence. PCS was also a transitional step between previous CFR variants and descendant algorithms that provided further efficiency and qualitative improvements.

- **Finding Optimal Abstract Strategies in Extensive-Form Games (AAAI 2012)**, [49]. In Chapter 4, we investigated the issues of abstraction pathologies and the overfitting effect. These issues plague attempts to approximate Nash equilibria by solving abstract games. **CFR-BR** is the first tractable game solving algorithm that solves abstract games while avoiding these issues, by provably converging to the least exploitable strategy that can be represented within an abstraction.

- **Evaluating State-Space Abstractions in Extensive-Form Games (AAMAS 2013)**, [52]. In Chapter 5, we explored the task of **creating and evaluating abstract games**. We demonstrated how abstractions could be evaluated through their ability to represent a Nash equilibrium, provided strong evidence supporting the use of imperfect recall abstractions, and revealed the powerful abstraction techniques we have used in the Annual Computer Poker Competition.

- **Data Biased Robust Counter Strategies (AISTATS 2009)**, [51]. In Chapter 6 we investigated the offline opponent modelling and counter-strategy task. By computing robust counter-strategies to use against a flawed opponent, we aim to outperform a Nash equilibrium strategy while retaining most of its worst-case guarantees. The **Data Biased Response** algorithm achieves this tradeoff while using observations of the opponent's strategy, avoiding the suboptimal behaviour of the earlier Restricted Nash Response algorithm.

- **Strategy Evaluation in Extensive Games with Importance Sampling (ICML 2008)**, [19]. In Chapter 7 we explored the **Imaginary Observations** technique for low-variance unbiased value estimation. This technique can be used in online and offline settings, and can also be used on-policy and off-policy. In Chapter 8, we used this technique to analyse the 2007 and 2008 Man-vs-Machine Poker Championships, showing for the first time that Polaris earned its 2008 victory. The Imaginary Observations technique was also a key component of this 2008 victory, when Polaris used its online off-policy form to dynamically change strategies during each match.

- **Heads-up Limit Hold'em Poker is Solved (Science, 2015)** [17]. Finally, in Chapter 9 we presented our recent milestone result of **essentially solving heads-up limit Texas hold'em**. This was the first human-scale imperfect information game to be solved. While we have pursued the goal of solving this game for more than a decade, it was finally made possible by the development of CFR+, a new variant of the Counterfactual Regret Minimization algorithm.

10.2 Recent Developments

The papers presented in this thesis span from 2008 to 2015. Ongoing research has extended this work in exciting ways. In this section, I will briefly describe recent developments in two aspects of this research.

10.2.1 CFR's Flexibility

In Chapters 3, 4, and 9, we described the Counterfactual Regret Minimization algorithm and several of its variants, such as PCS, CFR-BR, and CFR+. In these papers we focussed on two-player zero-sum games, and while we explored the use of imperfect recall abstractions in Chapter 5, most of our papers focus on perfect recall games and abstractions. This focus was chosen because CFR is only guaranteed to converge to a Nash equilibrium in the two-player zero-sum perfect recall setting.

One of CFR's most exciting properties, however, is that it remains well-defined outside of these bounds. In practice, we have obtained excellent empirical results by applying CFR to multiplayer games [79; 33; 32], non-zero-sum games [47; 32], and imperfect recall games [104; 52], including games that involve all three at once.

The application to multiplayer games is particularly interesting. Recently, Szafron and Gibson have used CFR to solve the game of 3-player Kuhn poker [91], producing a set of equations that describe a family of Nash equilibria. In general, however, we have found that CFR does not converge to a Nash equilibrium even in slightly larger games such as 3-player Leduc hold'em. This is not necessarily a disadvantage, as even if a Nash equilibrium could be computed, it would offer no performance guarantees to a single agent in a multiplayer game. And, even though it does not converge to a Nash equilibrium, CFR does appear to produce strong strategies in practice for large multiplayer games. In the Annual Computer Poker Competition, our 3-player limit entries from 2009 onwards have been created by running CFR. The resulting strategies have won each year's competition by a large margin, and yet little is understood about what properties are making them effective. Discovering what properties CFR and "CFR Strategies" might have in a multiplayer game is an exciting direction for future research.

10.2.2 Implicit Modelling

In Chapters 6 and 7, we explored aspects of offline and online opponent modelling and adaptation. Bard has recently extended this work into the Implicit Modelling framework [8]. DBR is an example of an explicit modelling approach: we attempt to model the opponent's entire strategy, and then compute a counter-strategy. To be effective, this approach requires a large amount of data. Bard's implicit modelling framework reverses the process: we start by computing a portfolio of counter-strategies that we believe will be generally useful against the opponents we are likely to face, and then use the Imaginary Observations technique to choose amongst them during a match. This approach creates an **implicit model** of the

opponent that is much easier to learn: a vector representing the expected value of using each strategy in our portfolio against the opponent being faced.

10.3 Future Work

The contributions presented in this thesis have advanced the state-of-the-art across the full scope of the Abstraction-Solving-Translation procedure. And although we have now solved the simplest of the human-scale poker games, this field of research is still in its early days. In this section, I will note several promising areas for future work. Although there are many exciting directions to explore in the poker domain, particularly as attention shifts towards the no-limit Texas hold'em variant that is most popular amongst humans, I will focus on general extensions of this work.

10.3.1 Game Solving

10.3.1.1 CFR+

The CFR+ algorithm described in Chapter 9 represents a significant step in the development of the CFR family of algorithms. While we have recently published a paper proving that it is theoretically guaranteed to converge [95], many aspects of CFR+ are not yet fully understood. For example, the current theoretical convergence bounds for CFR+ are the same as for CFR, while in practice CFR+ converges dramatically faster. Further, it is not yet known if the current strategy is guaranteed to converge, even though it does so quite reliably in our experiments.

With our current understanding, CFR+ is also difficult to use with abstraction. The root cause is that CFR+ does not perform well when sampling techniques are used. In our experiments with Monte Carlo variants such as CS and PCS, sampled CFR+ is outperformed by sampled CFR. For extremely large abstract games, such as no-limit Texas hold'em abstractions, card abstraction techniques reduce the memory needed to represent a strategy but not the computational cost to traverse the tree. Sampling techniques make it possible to quickly traverse parts of a game tree while updating many abstract information sets.

If a combination of CFR+ and sampling can be found that converges quickly, then it may be feasible to use CFR+ in place of CFR for solving very large abstract games. This may greatly improve our ability to create agents for domains that are larger and more challenging than HULHE, such as no-limit Texas hold'em.

10.3.1.2 Nash Equilibrium Refinements

While Nash equilibria are guaranteed to do no worse than tie in two player zero-sum games, they can still make mistakes. For example, it is not contradictory for a Nash equilibrium to make seemingly simple mistakes, such as folding the strongest possible hand (the "nuts") in a poker game. These errors can occur in subgames following an action that it is unprofitable for the opponent to take, and cannot be so

egregious as to cause the opponent to start taking the action. We call these "unexploitable errors", as their existence does not increase the value of a best response. And yet, a suboptimal opponent who does choose the unprofitable action will not be punished for their choice.

These have been observed in practice, particularly in strategies for no-limit Texas hold'em. Solutions to abstract no-limit games typically do not go all-in as the first action, as this is less profitable than playing the game normally. We have found that following this untaken all-in bet, the solutions tend to have a suboptimally wide calling range: they will call the all-in bet with too many of their weak hands. And while the optimal strategy is still to play the game normally, doing so requires great skill and precision. A suboptimal opponent may not be able to do so, but can very easily go all-in and make the remainder of the game almost a coin flip.

Ideally, we would like to compute a strategy that avoids such errors. Fortunately, there are solution concepts that are refinements of the Nash equilibrium concept, such as quasi-perfect equilibrium or sequential equilibrium. These solution concepts maximize expected value at every information set, instead of maximizing expected value at the root.

What is currently missing is an algorithm for computing such strategies that is as practical and efficient as CFR. While CFR can be combined with "trembling hand" approaches to compute a trembling hand equilibrium, this somewhat impacts CFR's practical speed as cutoffs are eliminated, and it is currently unknown how long CFR would take to converge at each information set. If a CFR variant (or other algorithm) can be found that efficiently converges to one of these Nash equilibrium refinements, then the resulting strategies should increase their winnings against suboptimal opponents.

10.3.2 Opponent Modelling

In Chapter 6, we discussed the Data Biased Response algorithm for computing robust counter-strategies. DBR is a practical algorithm that generates effective tradeoffs between robustness and exploitation. Unlike the earlier Restricted Nash Response algorithm, it can reliably use models constructed from observed data, instead of requiring the complete opponent strategy.

However, there is considerable room for improvement in the offline opponent modelling task. Note that DBR is not tied to the frequency-count models that we used in the paper. Instead, it can be combined with any opponent modelling technique that can provide both a prediction of the opponent's strategy and a confidence in that prediction at each information set. In particular, the frequency count model that we used suffers from data sparsity. It requires a large number of observed hands to fill in the opponent model, and does not attempt to generalize from observations to fill in unobserved or under-observed parts of the model. By combining DBR with a more sophisticated machine learning technique that attempts to generalize from observations, it may be possible to create more effective counter-strategies while using a more reasonable quantity of data.

10.4 Concluding Remarks

The central challenge of artificial intelligence research is to create computer programs that can independently consider their environment, learn about its parameters, and choose actions so as to reach their goals. Games have a long history as a testbed for artificial intelligence research because they offer many types of challenges. Games provide complex yet well-defined state spaces to act within, clear goals and metrics for measuring success, and the opportunity for comparisons against humans amateurs and experts: the only other source of intelligent agents against which our artificially intelligent agents can be compared. By pursuing the goal of creating strong game playing agents, we can learn general techniques that can be applied to artificially intelligent agents that act in real-world domains.

Different game domains focus on different components of this task. Following successes in the deterministic perfect information games of chess and checkers, the go research community continues in a domain with a large branching factor, where alpha-beta search is less effective. The real-time-strategy game research community, typified by the Starcraft AI challenge [24], stresses the importance of making real-time actions in an extremely large and imperfect information space. The General Game Playing research community [31] investigates the creation of agents that can learn to play arbitrary board games on their own, without intervention by humans. In a similar vein, the Arcade Learning Environment community aims to create computer agents that can learn to play arbitrary, unseen Atari 2600 games [10; 68].

The central focus of computer poker research is to discover effective algorithms for learning in large stochastic and imperfect information domains with multiple other agents with varying abilities. Even more so than the classic artificial intelligence domains of chess and checkers, we believe that the poker domain has much in common with the real world scenarios that we would like virtual agents and physical robots to act in. Agents in real world scenarios must act while coping with imperfect information, stochastic outcomes, and the presence of other agents with their own goals and abilities.

Through our research in the poker domain, we aim to make progress towards the full scope of real world challenges. And, returning to the beginning of game theory, John von Neumann drew a similar connection. As we noted in the introduction, when asked about perfect information games, he replied:

> "Real life is not like that. 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."
> – John von Neumann [20]

Bibliography

[1] Poker: A big deal. *The Economist*, December 22:31–38, 2007.

[2] Victor Allis. A knowledge-based approach to connect-four. the game is solved: White wins. Master's thesis, Vrije Universiteit, 1988.

[3] Victor L. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, University of Limburg, 1994.

[4] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, 2007.

[5] R.J. Aumann. game theory. In Steven N. Durlauf and Lawrence E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, Basingstoke, 2008.

[6] Charles Babbage. *Passages from the Life of a Philosopher*. Longman, Green, Longman, Roberts, and Green, London, 1864.

[7] Nolan Bard, Michael Johanson, and Michael Bowling. Asymmetric abstractions for adversarial settings. In *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-14)*, 2014. http://webdocs.cs.ualberta.ca/~games/poker/publications/2014-aamas-asymmetric-abstractions.pdf.

[8] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*, 2013. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAMAS13-modelling.pdf.

[9] Nolan Bard, Deon Nicholas, Csaba Szepesvri, and Michael Bowling. Decision-theoretic clustering of strategies. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-15)*, 2015. http://poker.cs.ualberta.ca/publications/2015-AAMAS-response-clustering.pdf.

[10] M.G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[11] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, 2003. http://webdocs.cs.ualberta.ca/~games/poker/publications/IJCAI03.pdf.

[12] Darse Billings. Computer poker. Master's thesis, University of Alberta, 1995. http://webdocs.cs.ualberta.ca/~games/poker/publications/billings.msc.pdf.

[13] Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron. The challenge of poker. volume 134, 2002. http://webdocs.cs.ualberta.ca/~games/poker/publications/AIJ02.pdf.

[14] Darse Billings and Morgan Kan. A tool for the direct assessment of poker decisions. *The International Computer Games Association Journal*, 29(3):119–142, 2006. http://webdocs.cs.ualberta.ca/~games/poker/publications/divat-icgaj.pdf.

[15] Émile Borel and Jean Ville. *Applications de la théorie des probabilités aux jeux de hasard*. Gauthier-Villars, 1938.

[16] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. The cepheus website. http://poker.srv.ualberta.ca, 2015.

[17] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, January 2015. http://webdocs.cs.ualberta.ca/~games/poker/15science.html.

[18] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved: Supplemental online material. http://johanson.ca/publications/poker/2015-science-hulhe/2015-science-hulhe-supplement.pdf, 2015.

[19] Michael Bowling, Michael Johanson, Neil Burch, and Duane Szafron. Strategy evaluation in extensive games with importance sampling. In *Proceedings of the 25th Annual International Conference on Machine Learning (ICML-08)*, 2008. http://webdocs.cs.ualberta.ca/~games/poker/publications/ICML08.pdf.

[20] Jacob Bronowski. Ascent of man. Documentary, 1973. Episode 13.

[21] Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence (AAAI-14)*, 2014. http://johanson.ca/publications/poker/2014-aaai-cfrd/2014-aaai-cfrd.pdf.

[22] Michael Buro. The othello match of the year: Takeshi Murakami vs. Logistello. *International Computer Chess Association Journal*, 20(3):189–193, 1997.

[23] Michael Buro. Solving the oshi-zumo game. *Advances in Computer Games*, 135:361–366, 2004.

[24] Michael Buro and David Churchill. 2014 starcraft ai competition website. http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/, 2014.

[25] Murray Campbell, A. Joseph Hoane Jr., and Feng-hsiung Hsu. Deep blue. *Artificial Intelligence*, 134:57–83, January 2002.

[26] Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. In *Proceedings of the Twenty-Sixth Conference on Advances in Neural Information Processing Systems (NIPS-2012)*, 2012. http://webdocs.cs.ualberta.ca/~bowling/papers/12nips-kofn.pdf.

[27] M. Craig. *The Professor, the Banker, and the Suicide King: Inside the Richest Poker Game of All Time*. Grand Central Publishing, 2006.

[28] Charles Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, 2003.

[29] D.A. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, May 2012.

[30] N. Gatti. Extending the alternating-offers protocol in the presence of competition: Models and theoretical analysis. *Annals of Mathematics in Artificial Intelligence*, 55(3-4):189–236, April 2008.

[31] Michael Genesereth and Michael Thielscher. *General Game Playing*. Morgan & Claypool, 2014.

[32] Richard Gibson. Regret minimization in non-zero-sum games with appliations to building champion multiplayer computer poker agents. *arXiv*, 2013, 1305.0034v1. arxiv.org/abs/1305.0034.

[33] Richard Gibson and Duane Szafron. On strategy stitching in large extensive form multiplayer games. In *Proceedings of the Twenty-Fifth Conference on Neural Information Processing Systems (NIPS 2011)*, 2011. http://webdocs.cs.ualberta.ca/~games/poker/publications/NIPS11.pdf.

[34] Andrew Gilpin, Samid Hoda, Javier Peña, and Tuomas Sandholm. Gradient-based algorithms for finding nash equilibria in extensive form games. In *3rd International Workshop on Internet and Network Economics (WINE'07)*, 2007. http://www.cs.cmu.edu/~gilpin/papers/egt.wine07.pdf.

[35] Andrew Gilpin and Tuomas Sandholm. Optimal rhode island hold'em poker. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1684–1685, 2005. http://www.cs.cmu.edu/~sandholm/RIHoldEm.ISD.aaai05proceedings.pdf.

[36] Andrew Gilpin and Tuomas Sandholm. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006. http://www.cs.cmu.edu/~gilpin/papers/texas.aaai06.pdf.

[37] Andrew Gilpin and Tuomas Sandholm. Lossless abstraction of imperfect information games. *Journal of the ACM*, 54(5), 2007. http://www.cs.cmu.edu/~gilpin/papers/extensive.JACM.pdf.

[38] Andrew Gilpin and Tuomas Sandholm. Expectation-based versus potential-aware automated abstraction in imperfect information games: An experimental comparison using poker. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2008. http://www.cs.cmu.edu/~gilpin/papers/abscomp.aaai08.pdf.

[39] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2007. http://www.cs.cmu.edu/~gilpin/papers/gs3.aaai07.pdf.

[40] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[41] Samid Hoda, Andrew Gilpin, Javier Peña, and Tuomas Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010. http://www.cs.cmu.edu/~sandholm/proxtreeplex.MathOfOR.pdf.

[42] Andrew Hodges. *Alan Turing: The Enigma*. Vintage, 1992.

[43] F. H. Hsu. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press, 2002.

[44] Eric Jackson. The Annual Computer Poker Competition webpage. http://www.computerpokercompetition.org, 2012.

[45] Eric Jackson. Slumbot: An implementation of counterfactual regret minimization on commodity hardware. In *2012 Computer Poker Symposium*, 2012.

[46] Eric Jackson. Slumbot nl: Solving large games with counterfactual regret minimization using sampling and distributed processing. In *The AAAI-13 Workshop on Computer Poker and Imperfect Information*, 2013.

[47] Michael Johanson. Robust strategies and counter-strategies: Building a champion level computer poker player. Master's thesis, University of Alberta, 2007. http://webdocs.cs.ualberta.ca/~games/poker/publications/johanson.msc.pdf.

[48] Michael Johanson. Measuring the size of large no-limit poker games. Technical Report TR13-01, Department of Computing Science, University of Alberta, 2013. http://webdocs.cs.ualberta.ca/~games/poker/publications/2013-techreport-nl-size.pdf.

[49] Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive-form games. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAAI12-cfrbr.pdf.

[50] 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 Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12)*, pages 837 – 844, 2012. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAMAS12-pcs.pdf.

[51] Michael Johanson and Michael Bowling. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS-09)*, 2009. http://webdocs.cs.ualberta.ca/~games/poker/publications/AISTATS09.pdf.

[52] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*, 2013. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAMAS13-abstraction.pdf.

[53] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 258–265, 2011. http://webdocs.cs.ualberta.ca/~games/poker/publications/ijcai2011_accelerated_best_response.pdf.

[54] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *NIPS*, 2008. http://webdocs.cs.ualberta.ca/~games/poker/publications/NIPS07-rnash.pdf.

[55] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[56] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4(4):528–552, 1992.

[57] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14(2), 1996.

[58] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. Fast algorithms for finding randomized strategies in game trees. In *Annual ACM Symposium on Theory of Computing, STOC'94*, pages 750–759, 1994.

[59] Daphne Koller and Avi Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94:167–215, 1997.

[60] Harold W. Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.

[61] Marc Lanctot, Richard Gibson, Neil Burch, and Michael Bowling. No-regret learning in extensive-form games with imperfect recall. In *ICML*, 2012. http://mlanctot.info/files/papers/12icml-ir.pdf.

[62] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the Twenty-Third Conference on Advances in Neural Information Processing Systems (NIPS-2009)*, 2009. http://mlanctot.info/files/papers/nips09mccfr.pdf.

[63] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. Technical Report TR09-15, Department of Computing Science, University of Alberta, 2009. http://mlanctot.info/files/papers/nips09mccfr_techreport.pdf.

[64] Alessandro Lazaric, Jose Enrique Munoz de Cote, and Nicola Gatti. Reinforcement learning in extensive form games with incomplete information: the bargaining case study. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2007.

[65] Michael Littman and Martin Zinkevich. The aaai computer poker competition. *Journal of the International Computer Games Association*, 29, 2006.

[66] Peter McCracken and Michael Bowling. Safe strategies for agent modelling in games. In *AAAI Fall Symposium on Artificial Multi-agent Learning*, 2004. http://webdocs.cs.ualberta.ca/~bowling/papers/04aaai-fallsymp.pdf.

[67] Philip Mirowski. What were von neumann and morgenstern trying to accomplish? In Weintraub, editor, *Toward a History of Game Theory*, pages 113–147. Duke University Press, 1992. Mirowski cites Turing as author of the paragraph containing this remark. The paragraph appeared in [97], in a chapter with Turing listed as one of three contributors. Which parts of the chapter are the work of which contributor, particularly the introductory material containing this quote, is not made explicit.

[68] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[69] Oskar Morgenstern. The cold war is cold poker. *New York Times Magazine*, pages 21–22, February 5 1961.

[70] Hala Mostafa, Victor Lesser, and Gerome Miklau. Self-interested database managers playing the view maintenance game. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.

[71] mykey1961. Win rate with optimal strategy against limit raise bot. Two Plus Two Poker Forums, September 2007. http://forumserver.twoplustwo.com/15/poker-theory/win-rate-optimal-strategy-against-limit-raise-bot-2332/index6.html#post251612, retrieved April 14, 2011.

[72] J. F. Nash and L. S. Shapley. A simple 3-person poker game. In *Contributions to the Theory of Games I*, pages 105–116. Princeton University Press, 1950.

[73] Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):233–249, 2005.

[74] University of Alberta Computer Poker Research Group. The First Man-Machine Poker Championship webpage. http://www.computerpokercompetition.org/2007/, 2007.

[75] University of Alberta Computer Poker Research Group. The Second Man-Machine Poker Competition webpage. http://www.computerpokercompetition.org/, 2008.

[76] Martin Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[77] Ariel D. Procaccia and Jeffrey S. Rosenschein. Extensive-form argumentation games. In *The Third European Workshop on Multi-Agent Systems (EUMAS)*, 2005.

[78] Julie Rehmeyer, Nathan Fox, and Renzi Rico. Ante up, human: The adventures of polaris the poker-playing robot. *Wired*, 16.12:186–191, December 2008.

[79] Nick Abou Risk and Duane Szafron. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, 2010. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAMAS10.pdf.

[80] I. V. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[81] I. V. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

[82] John W. Romein and Henri E. Bal. Solving awari with parallel retrograde analysis. *IEEE Computer*, 36(26):26–33, 2003.

[83] Jonathan Rubin and Ian Watson. Computer poker: A review. *Artificial Intelligence*, 175(5-6):958–987, 2011. http://www.jonathan-rubin.com/files/CPReviewPreprintAIJ.pdf.

[84] Tuomas Sandholm. The state of solving large incomplete-information games, and application to poker. *AI Magazine*, 31(4):13–32, 2010. http://www.cs.cmu.edu/~sandholm/solving%20games.aimag11.pdf.

[85] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Muller, Rob Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 2007.

[86] Jonathan Schaeffer, Robert Lake, Paul Lu, and Martin Bryant. CHINOOK: The world man-machine checkers champion. *AI Magazine*, 17(1):21–29, 1996.

[87] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine, Series 7*, 41(314):256–275, March 1950.

[88] Brian Sheppard. World-championship-caliber Scrabble. *Artificial Intelligence*, 134:241–275, 2002.

[89] Jiefu Shi and Michael L. Littman. Abstraction methods for game theoretic poker. In *Revised Papers from the Second International Conference on Computers and Games*, CG '00, pages 333–345, London, UK, UK, 2002. Springer-Verlag. http://www.cs.rutgers.edu/~mlittman/attweb/papers/cg00-poker.ps.

[90] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 2005. http://webdocs.cs.ualberta.ca/~games/poker/publications/UAI05.pdf.

[91] Duane Szafron, Richard Gibson, and Nathan Sturtevant. A parameterized family of equilibrium profiles for three-player kuhn poker. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-13)*, 2013. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAMAS13-3pkuhn.pdf.

[92] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[93] Oskari Tammelin. Oskari Tammelin's cfr website. http://www.jeskola.net/cfr, 2014.

[94] Oskari Tammelin. Solving large imperfect information games using CFR+. *CoRR*, abs/1407.5042, 2014.

[95] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit texas hold'em. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015. http://webdocs.cs.ualberta.ca/~games/poker/publications/2015-ijcai-cfrplus.pdf.

[96] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[97] Alan Turing. Digital computers applied to games. In Bertram Vivian Bowden, editor, *Faster Than Thought*, chapter 25. Pitman, 1976.

[98] A.M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[99] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[100] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, Second Edition, 1947.

[101] Kevin Waugh. Abstraction in large extensive games. Master's thesis, University of Alberta, 2009. http://webdocs.cs.ualberta.ca/~games/poker/publications/waugh.msc.pdf.

[102] Kevin Waugh and J. Andrew Bagnell. A unified view of large-scale zero-sum equilibrium computation. In *AAAI Workshop on Computer Poker and Imperfect Information*, 2015.

[103] Kevin Waugh, David Schnizlein, Michael Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In *AAMAS*, 2009. http://webdocs1.cs.ualberta.ca/~games/poker/publications/AAMAS09-abstraction.pdf.

[104] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael Bowling. A practical use of imperfect recall. In *SARA*, 2009. http://webdocs.cs.ualberta.ca/~games/poker/publications/sara09.pdf.

[105] Wikipedia. Earth mover's distance — Wikipedia, the free encyclopedia, 2013.

[106] Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth International Congress Mathematics*, pages 501–504, Cambridge, 1913. Cambridge University Press.

[107] Martin Zinkevich, Michael Bowling, Nolan Bard, Morgan Kan, and Darse Billings. Optimal unbiased estimators for evaluating agent performance. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 2006. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAAI06.pdf.

[108] Martin Zinkevich, Michael Bowling, and Neil Burch. A new algorithm for generating equilibria in massive zero-sum games. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, 2007. http://webdocs.cs.ualberta.ca/~games/poker/publications/AAAI07-smallbot.pdf.

[109] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. Technical Report TR07-14, Department of Computing Science, University of Alberta, 2007. http://webdocs.cs.ualberta.ca/~games/poker/publications/NIPS07-cfr.pdf.

[110] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *NIPS*, 2008. http://webdocs.cs.ualberta.ca/~games/poker/publications/NIPS07-cfr.pdf.

Chapter 11

Appendix

Each chapter in this thesis presented one complete research paper, as it originally appeared. In this appendix, we will supply additional information and results for some of these papers that supplement the original results, or correct weaknesses that were discovered after publication.

11.1 Chapter 2: Worst-Case Evaluation

In Table 2.2, we presented a table comparing several ACPC competitors in terms of exploitability and in-game performance against Rockhopper, the winner of the 2010 ACPC. In Table 11.1, we supplement this table by also showing the in-game performance against PULPO (winner of the 2010 Total Bankroll event) and against Hyperborean.IRO, the least exploitable agent of those evaluated from the 2010 ACPC.

In 2012, I ran a followup experiment for the Computer Poker Symposium at the AAAI 2012 conference. In this second experiment, I again offered to all ACPC competitors to measure the exploitability of their agents. The intent was to measure the progress of the community between the 2010 and 2012 competitions. In the 2012 ACPC, Eric Jackson's agent Slumbot [45] was both the least exploitable agent and the winner of the Instant Runoff and Total Bankroll events.

Name	vs (1)	vs (4)	vs (6)	Exploitability
(1) Hyperborean.IRO	0	-3 $\pm$ 2	2 $\pm$ 4	135.427
(2) Hyperborean.TBR	3 $\pm$ 4	-1 $\pm$ 4	9 $\pm$ 4	141.363
(3) GGValuta	3 $\pm$ 6	-7 $\pm$ 2	1 $\pm$ 5	237.330
(4) Rockhopper	3 $\pm$ 4	0	7 $\pm$ 5	300.032
(5) GS6.IRO	-31 $\pm$ 5	-37 $\pm$ 6	-32 $\pm$ 6	318.465
(6) PULPO	-2 $\pm$ 4	-9 $\pm$ 2	0	399.387
(7) Littlerock	-70 $\pm$ 5	-77 $\pm$ 5	-125 $\pm$ 5	421.850

Table 11.1: Extended exploitability results for 2010 ACPC heads-up limit competitors.

Name	vs Slumbot	Exploitability
Slumbot	0	90.873
Hyperborean.IRO	-4 $\pm$ 3	106.035
ZBot	-21 $\pm$ 4	249.384
Littlerock	-18 $\pm$ 3	324.216

Table 11.2: Exploitability results for 2012 ACPC heads-up limit competitors.

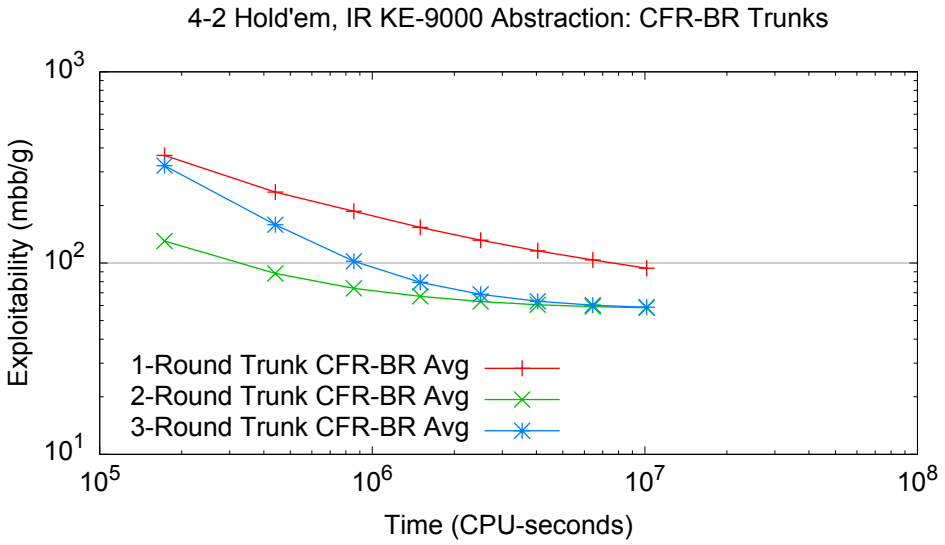


Figure 11.1: CFR-BR with 1-, 2-, and 3-round trunks. The real game is 4-round, 2-bet heads-up limit hold'em, and the abstraction is IR KE-9000, as described in Chapter 5.

11.2 Chapter 4: Optimal Abstract Strategies

In Chapter 4, we demonstrated the convergence rate of CFR-BR in heads-up limit Texas hold'em using a 1-round and 2-round trunk. While a 3-round trunk is possible, the memory cost is considerably higher (see Table 4.1), while the 2-round trunk nicely balances the size of each part of the game. Nonetheless, it is still interesting to compare the convergence rate, to see if the extra memory cost for a 3-round trunk is accompanied by an increase in speed.

We investigate this in Figure 11.1. To avoid the huge memory cost of a 3-round trunk in heads-up limit Texas hold'em, we ran this experiment in [4-round, 2-bet] heads-up limit hold'em, which simply limits the bets and raises in each round to two instead of four. In this figure, we see that the 2-round trunk converges more quickly than the 1-round trunk at all datapoints, matching our earlier results. In this game, the three round trunk offers no advantage over the two round trunk: it initially converges more slowly, and at the end exactly matches the two round trunk's performance.

In Figure 4.12, we demonstrated the convergence of CFR-BR in the abstraction we used for our 2011 ACPC heads-up limit agent. The resulting strategy reached an exploitability of 37.170 mbb/g, which at the time was by far the least exploitable

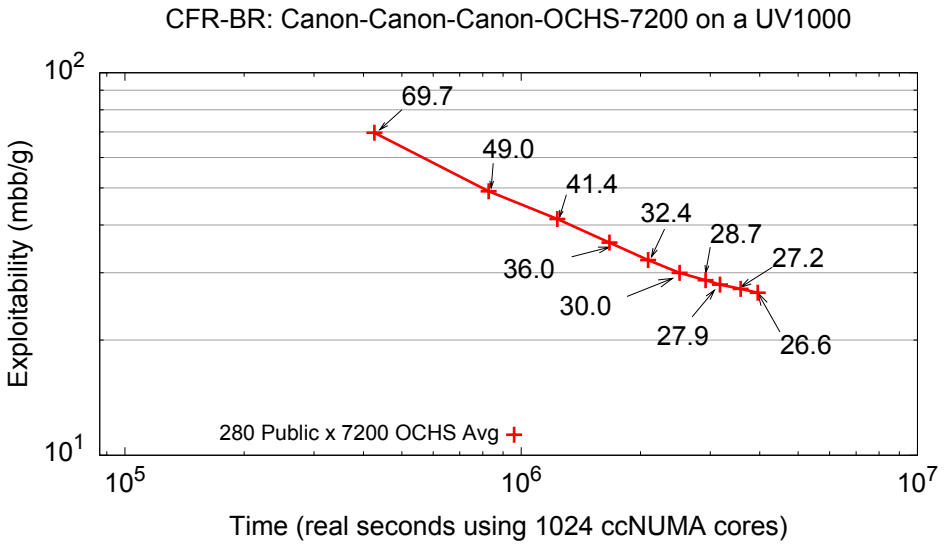


Figure 11.2: CFR-BR applied to a very large abstraction of heads-up limit Texas hold'em.

strategy ever produced: the best previous strategy was created by running CFR on the same abstraction, which reached an exploitability of 104 mbb/g. At the time, CFR-BR seemed like it might be a feasible way to get close enough to equilibrium to declare the game solved: if our abstraction was a sufficiently good model of the real game, and if we ran CFR-BR long enough, we might eventually reach our target of 1 mbb/g. In particular, we hoped that if the first three rounds used a lossless abstraction and only the river used lossy abstraction techniques, that we might reach our goal.

In pursuit of that goal, in 2012 we ran a second large experiment shown in Figure 11.2. The abstraction was canonical in the first three rounds. In the final round, the abstraction divided the public cards into 280 public buckets, and then subdivided each into (on average) 7200 imperfect recall OCHS buckets, with just over two million river buckets total. Solving this game took approximately two terabytes of RAM on an SGI UV1000 ccNUMA computer with 2048 cores, half of which were used for our experiment. When this abstraction was solved with CFR-BR, the strategy reached an exploitability of 26.6 mbb/g and may have eventually reached 25 mbb/g if we had continued running it. While this was a substantial improvement over our previous 37 mbb/g strategy, it convinced us that our abstraction technique, at our target game size, was not sufficient to reach 1 mbb/g.

11.3 Chapter 5: State-Space Abstraction

11.3.1 IR-*-PHS results

In our comparison of abstraction techniques in Tables 5.3, 5.4 and 5.5, we focussed on perfect recall abstractions with a branching factor of 10 buckets per round and imperfect recall abstractions with 9000 buckets per round, which are approximately

the same size. However, after the publication of the paper, Marcin Dukaczewski[1] contacted me to note that this comparison was unfair to the imperfect recall percentile hand strength combination, as there are only 1950 possible $E[HS]$ values on the river. In this section, we will present additional figures that address this flaw, and further investigate the relative strength of abstract strategies as the abstraction size is varied.

Using our abstraction implementation, in the IR-*-PHS abstractions only 1950 of the 9000 river buckets were used and the remaining 7050 were simply empty. A fairer comparison of the IR-*-PHS abstractions would use at most 1950 nonempty river buckets, and redistribute the remaining river buckets to the flop and turn. As we illustrated in Table 5.2, the cost of each river bucket is equal in memory cost to 9 turn buckets or 81 flop buckets, allowing us to greatly improve the quality of the abstraction in those rounds while retaining the same overall game size. Of course, if this movement of buckets proved advantageous for the IR-*-PHS family of abstractions, then it might also be advantageous for the IR-*-KO family.

In Table 11.3, we present a new crosstable of in-game performance using strategies generated by PCS. The original imperfect recall strategies, which used the same number of buckets in the flop, turn and river, remain with the IR-EQ name prefix (EQ for "equal" number of buckets on each round). The IR-FT (FT for "Flop/Turn") name prefix is used for the new abstractions, which have 1950 river buckets and 66105 flop and turn buckets. In the case of the IR-FT-PHS-PHS strategies, a nested abstraction was used in the flop and turn, which used $406E[HS^2] \times 103E[HS] = 66178$ buckets, slightly larger than the target of 66105. The conclusions reached in the original paper are unchanged: the IR-EQ-KE-KO, IR-FT-KE-KO, and IR-FT-KO-KO abstractions have the highest overall winnings, do not lose to any other agents, and are inseperable from each other.

In Table 11.4, we also compare these new strategies in terms of exploitability, using both CFR and CFR-BR. While the new IR-FT-*-PHS strategies improve as expected over the (unfairly compared) IR-EQ-*-PHS variants, the improvement is not sufficient to match the IR-EQ-KE-KO or IR-FT-KO-KO strategies, which remain the strongest overall in terms of CFR-BR exploitability.

In Figure 11.3, we compare these families of abstractions by varying the number of buckets in the abstraction. The strategies evaluated were generated by CFR-BR. The smallest IR abstraction is quite small, at just 100 buckets per round. Overall, the IR-EQ-KE-KO and IR-FT-KE-KO strategies were less exploitable than the other variants at all tested abstraction sizes.

[1]Thank you to Marcin Dukaczewski for noting this flaw in our analysis.

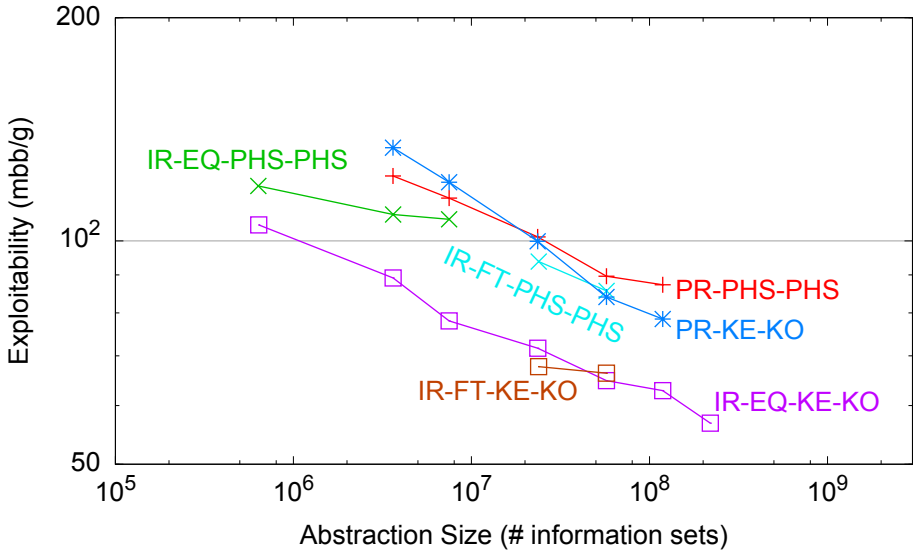


Figure 11.3: Exploitability of CFR-BR strategies in six types of abstractions as the abstraction size is varied.

	Opponents																		
	PR-PHS-PHS	PR-PHS-KO	PR-KE-PHS	PR-KE-KO	PR-KO-PHS	PR-KO-KO	IR-EQ-PHS-PHS	IR-EQ-PHS-KO	IR-EQ-KE-PHS	IR-EQ-KE-KO	IR-EQ-KO-PHS	IR-EQ-KO-KO	IR-FT-PHS-PHS	IR-FT-PHS-KO	IR-FT-KE-PHS	IR-FT-KE-KO	IR-FT-KO-PHS	IR-FT-KO-KO	Average
PR-PHS-PHS		0	-1	-2	14	14	-8	-14	-21	-25	-19	-22	-13	-18	-21	-26	-20	-25	-11
PR-PHS-KO	0		0	-2	14	13	-8	-14	-20	-26	-17	-22	-13	-19	-22	-26	-21	-25	-11
PR-KE-PHS	1	0		-2	15	14	-6	-12	-18	-24	-16	-21	-10	-16	-19	-24	-18	-24	-10
PR-KE-KO	2	2	2		16	16	-6	-12	-18	-24	-15	-21	-10	-16	-19	-23	-18	-23	-9
PR-KO-PHS	-14	-14	-15	-16		-1	-25	-30	-34	-39	-32	-37	-27	-32	-35	-40	-35	-40	-26
PR-KO-KO	-14	-13	-14	-16	1		-25	-30	-34	-39	-31	-37	-27	-32	-34	-40	-34	-40	-25
IR-EQ-PHS-PHS	8	8	6	6	25	25		-5	-15	-18	-10	-15	-7	-11	-15	-19	-14	-18	-3
IR-EQ-PHS-KO	14	14	12	12	30	30	5		-9	-14	-6	-10	-1	-5	-10	-13	-9	-12	1
IR-EQ-KE-PHS	21	20	18	18	34	34	15	9		-6	2	-3	10	3	-3	-7	-2	-7	8
IR-EQ-KE-KO	25	26	24	24	39	39	18	14	6		9	2	14	10	2	0	4	0	14
IR-EQ-KO-PHS	19	17	16	15	32	31	10	6	-2	-9		-7	7	1	-4	-10	-5	-10	6
IR-EQ-KO-KO	22	22	21	21	37	37	15	10	3	-2	7		11	7	1	-3	1	-3	11
IR-FT-PHS-PHS	13	13	10	10	27	27	7	1	-10	-14	-7	-11		-5	-13	-16	-11	-15	0
IR-FT-PHS-KO	18	19	16	16	32	32	11	5	-3	-10	-1	-7	5		-6	-11	-5	-10	5
IR-FT-KE-PHS	21	22	19	19	35	34	15	10	3	-2	4	-1	13	6		-5	0	-4	10
IR-FT-KE-KO	26	26	24	23	40	40	19	13	7	0	10	3	16	11	5		6	0	15
IR-FT-KO-PHS	20	21	18	18	35	34	14	9	2	-4	5	-1	11	5	0	-6		-5	10
IR-FT-KO-KO	25	25	24	23	40	40	18	12	7	0	10	3	15	10	4	0	5		14
Max	26	26	24	24	40	40	19	14	7	0	10	3	16	11	5	0	6	0	

Table 11.3: Crosstable of heads-up limit Texas hold'em CFR strategies in a variety of card abstractions. Each cell shows the score for the row player in milli-big-blinds/game (mbb/g) over a 10 million hand duplicate match. Results have a 95% confidence interval of 1.1 mbb/g. Strategies were trained using PCS CFR, run for 4 days on a 48-core 2.2 GHz AMD computer, which in practice is sufficient for the perfect recall strategies to converge to 1 mbb/g within their abstraction.

	CFR			CFR-BR		
	PR	IR-EQ 9000	IR-FT 66105-1950	PR	IR-EQ 9000	IR-FT 66105-1950
PHS-PHS	288.942	*358.188	320.862	89.632	*94.841	85.655
PHS-KO	289.152	318.197	286.415	90.371	85.275	80.746
KE-PHS	281.63	*339.872	325.034	90.720	*80.557	75.313
KE-KO	282.856	282.395	280.405	84.039	64.820	66.338
KO-PHS	335.902	*355.881	317.113	105.389	*88.546	80.355
KO-KO	330.319	291.574	274.701	105.523	73.091	69.340

Table 11.4: Exploitability of CFR-BR agents in tested abstractions. These results are measured in milli-big-blinds/game (mbb/g) and are exact. Entries marked with a * are unfair comparisons of the IR-*-PHS feature as they use more than 1950 river buckets, and are kept for consistency with our earlier published results. In such cases, the adjacent IR-FT values should be considered instead, which redistribute these buckets to the flop and turn.

11.3.2 Canonical Preflop and Flop Results

One of the common abstraction choices that was not investigated in the abstraction paper is the use of "canonical flop" abstractions. By reducing the number buckets in later rounds, we can add a far greater number of buckets to earlier rounds. This makes it quite feasible to allocate the 1,286,792 flop buckets necessary for a canonical (*i.e.* lossless) flop abstraction, in which every canonical combination of flop cards and preflop cards are represented separately.

This leads to an interesting tradeoff. How many turn and river buckets should we be willing to sacrifice in order to achieve this lossless flop representation? In Figure 11.4, we present a plot showing two abstraction families: IR-C-KE-KE-KO and IR-C-C-KE-KO. The naming convention differs slightly from what we used earlier by explicitly describing each of the four rounds, with C representing a canonical abstraction. The IR-C-KE-KE-KO abstractions used 100, 570, 1175, 3700, 9000, 18630 and 34470 buckets on the flop, turn and river, while IR-C-C-KE-KO used 570, 1175, 3700, 9000, 18630, 60k, 180k and 540k buckets on the turn and river. Aside from the first few small IR-C-C-KE-KO datapoints, the two curves almost perfectly overlap and follow the same slope. The IR-C-C-KE-KO abstraction with 3700 buckets and the IR-C-KE-KE-KO abstraction with 18630 buckets are both exploitable for 62.8 mbb/g, and the IR-C-C-KE-KO with 18630 buckets and the IR-C-KE-KE-KO with 34470 are also similar at 57.1 and 56.8 mbb/g respectively.

11.3.3 Asymmetric Abstractions

All of the abstractions in the original paper and in our extended analysis here use **symmetric abstractions** in which both players use the same size of game and abstraction technique. However, it is also interesting to consider **asymmetric abstractions**, in which one player's abstraction is finer-grained than their opponent's. When these games are solved, in comparison to a symmetric abstract strategy, the finer-grained player's strategy tends to be more exploitable but also more effective in one-on-one play against a wide range of weaker adversaries. The coarser-grained player's strategy tends to be less exploitable than its symmetric equivalent, but also weaker in one-on-one play. This mirrors our findings from CFR-BR, which is an extreme form of one player's abstraction being coarser than the opponent's. For a full treatment of this subject, see [7].

11.4 Chapter 6: Offline Modelling and Counter Strategies

Poker Academy Online was an online play-money poker training website, founded in part by alumni from the University of Alberta's Computer Poker Research Group, which operated until August 2011.[2] After the 2007 Man-vs-Machine Poker Championship, the University of Alberta had six rooms on Poker Academy where players could play for play money against Polaris and experimental agents that we had created. The players were aware that Polaris was the computer agent that had competed against human pros. This gave us a chance to benchmark its performance against non-professional opponents, and also to test opponent modelling and counter strategy techniques such as data biased response.

[2]The current website running at the old Poker Academy address offers a different service and is unrelated.

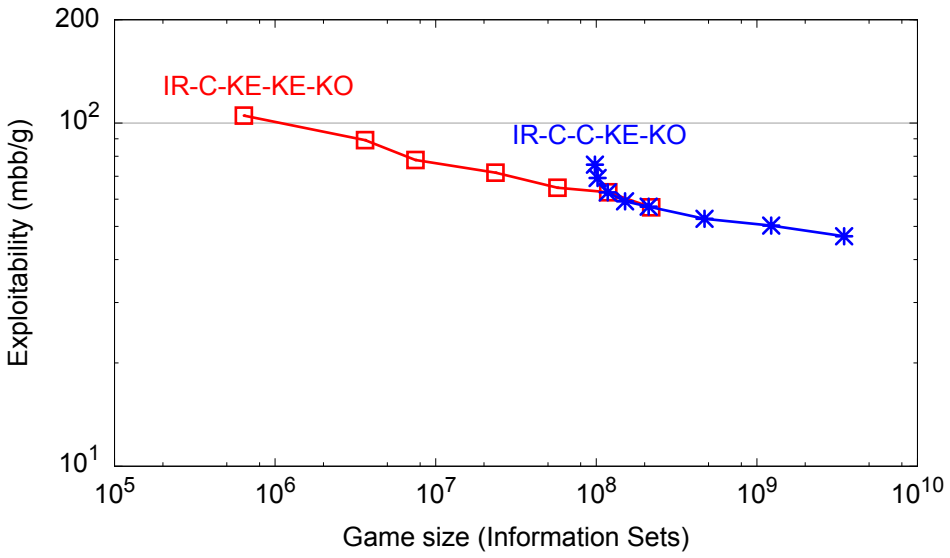


Figure 11.4: Exploitability of IR-KE-KO abstracted strategies with and without a canonical flop abstraction.

Agent	Games Played	Average Winnings
Polaris 2007 Pink	563461	145 ± 14
Polaris 2007 Orange	582026	155 ± 14
Polaris 2008	531598	120 ± 14
DBR-PokerAcademy	35839	341 ± 59

Table 11.5: Performance of several CPRG agents that played for play money on Poker Academy Online between 2007 and 2009.

Over their time on the website, our agents played over two million hands of heads-up limit Texas hold'em against these human opponents. In 2007, the Polaris "Pink" and "Orange" strategies that played in matches (1 and 4) and 2 respectively of the 2007 event were online. In 2008, they were replaced with Polaris 2008, which dynamically switched between five component strategies.

In 2009, we used Data Biased Response to create a counter-strategy to the humans that had played against Polaris. This strategy used the same 8-bucket perfect recall PHS abstraction that was used for Polaris 2007's Orange strategy. To create the opponent model, we used the one million games that had been played at that time against Polaris 2007 and 2008, treating all of the human opponents as being a single adversary. We then put this counter-strategy on Poker Academy, labelled as being an experimental strategy (*i.e.* not Polaris), for players to play against.

In Table 11.5, we compare the performance of these players in their matches on Poker Academy Online. Note that this was not a controlled experiment, as we had no way of evaluating whether the same population was playing against the Polaris strategies and the DBR strategy. Nonetheless, even though the DBR's card abstraction was far weaker than the abstraction used in Polaris 2008 (see Table 2.3), its winrate was far higher than Polaris', which was online at the same time in a different room. It is not surprising that amateur humans would have exploitable flaws, and the DBR strategy appears to have exploited them.