No man should escape our universities without knowing how little he knows.

– Robert Oppenheimer, 1904–1967.

University of Alberta

ABSTRACTION IN LARGE EXTENSIVE GAMES

by

Kevin Waugh

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Fall 2009
Edmonton, Alberta

Examining Committee

Michael Bowling, Computing Science

Dale Schuurmans, Computing Science

Nathan Sturtevant, Computing Science

Peter Hooper, Mathematics and Statistics

To my parents, Betty-Anne and Gerry,
and my sister, Jennifer.

Abstract

Extensive games model scenarios where multiple agents interact with a potentially stochastic environment. In the case of two-player zero-sum games, we have efficient techniques for computing solutions. These solutions are optimal in that they guarantee the player the highest expected reward against a worst-case opponent. Unfortunately, there are many interesting two-player zero-sum games that are too large for the state-of-the-art solvers. For example, heads-up Texas Hold'em has 10^{18} game states and requires over two petabytes of storage to record a single strategy[1]. A popular approach for tackling these large games is to use an abstraction technique to create a smaller game that models the original game. A solution to the smaller abstract game can be computed and is presumed good in the original game. A common assumption is that the more accurately the abstract game models the original game the stronger the resulting strategy will be. There is substantial empirical evidence to support this assumption and, prior to this work, it had not been questioned. In this thesis, we show that this assumption is not true. That is, using a larger, more accurate, abstract game can lead to weaker strategies. To show this, we must first formalize what it means to abstract a game as well as what it means for an abstraction to be bigger or more informed. We then show that the assumption fails to hold under two criteria for evaluating a strategy. The first is exploitability, which measures performance against a worst-case adversary. The second is a new evaluation criterion called the domination value, which essentially measures how many mistakes (actions that one should never take) a strategy makes. Despite these abstraction pathologies, we argue that solutions to the larger abstract games tend to make fewer mistakes. This leads us to develop a new technique, called strategy grafting, that can be used to augment a strong base strategy. It does this by decomposing an extensive game into multiple sub-games. These sub-games are then solved separately, but with shared knowledge of the underlying base strategy. This technique allows us to make effective use of much larger strategy spaces than previously possible. We provide a weak theoretical guarantee on the quality of this new technique, but we show in practice that it does quite well even when the conditions on our theoretical results do not hold. Furthermore, when evaluated under our new criterion, we notice that grafted strategies tend to make fewer mistakes than their base strategy.

[1] Estimated assuming four bytes per action while additionally taking advantage of suit isomorphisms.

Acknowledgements

I have enjoyed the time I have spent at the University of Alberta. The fun that I have had is certainly not due to Edmonton's weather, but instead to the colleagues and friends that I have met here. Without their influence, I would not be the person that I am today. I would like to give special thanks to the following people:

- **Michael Bowling** and **Dale Schuurmans**, my supervisors. Thank you both for the time, patience and guidance that you have afforded me. You both have gone far beyond what anyone could expect from a supervisor, especially considering your other commitments.
- **Duane Szafron** and **Jonathan Schaeffer**, my internship supervisors. As a naive undergraduate student I thought that professors sat in their office's and *researched* all day. You have both shown me that research is in fact hard, but very enjoyable and rewarding, work.
- **Piotr Rudnicki**, my coach. Joining the Programming Club was likely the best choice that I made during my undergraduate degree. It has changed the way that I think, vastly increased my knowledge and abilities, and provided me with tremendous opportunities to travel and network. Your persistence has done so much for me and is greatly appreciated.
- All past and present members of the **Computer Poker Research Group** and those involved with my research. Specifically, **Nick Abou Risk**, **Nolan Bard**, **Neil Burch**, **Johnny Hawkin**, **Michael Johanson**, **Morgan Kan**, **Bryce Paradis**, **Dave Schnizlein**, **Nathan Sturtevant** and **Marty Zinkevich**. Thank you all for the helpful discussions. This work would not be possible without your input.
- My friends: **Maria Cutumisu**, **Josh Davidson**, **Sam Bou Fakhreddine**, **Jenette Fernandez**, **Jeff Grajkowski**, **Andy Hiew**, **Adam Jocksch**, **Kurt** and **Chloe McMillan**, **Jennifer Miller**, **Thomas Mortimer**, **Clemens Park**, **Jordan Patterson**, **Jeff Siegel**, **Christian Smith**, **Mark Trommelen**, **Terry White** and **Seul Kee Yoon**. Thank you all for your support and for making Edmonton fun.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Extensive games provide a general model for describing the interactions of multiple agents within a potentially stochastic environment. They subsume other sequential decision making models such as finite horizon Markov decision processes, finite horizon partially observable Markov decision processes, and many other multi-agent scenarios such as stochastic games. This makes extensive games a powerful tool for representing a variety of complex situations. Moreover, it means that techniques for computing solutions to extensive games are a valuable commodity that can be applied in many different domains. The usefulness of the extensive game model is dependent on the availability of solution techniques that scale well with respect to the size of the model. Recent research, particularly motivated by the domain of poker, has resulted in significant developments in scalable solution techniques for two-player zero-sum extensive games. Here, by a solution we mean a strategy that provides the highest expected reward against a worst-case adversary. The classic linear programming techniques [13] can solve games with approximately 10^7 states [2] on modern machines. More recent techniques [6, 23], which can explicitly maintain the sparse structure of the problem, can solve games with over 10^{12} states.

Despite this dramatic improvement, there still remains many interesting extensive games whose solutions cannot be found due to current algorithmic and hardware limitations. Fixed-limit heads-up Texas Hold'em poker, the smallest poker game that is played competitively by humans, has 10^{18} [2] states and would require over 2 petabytes of storage [12] to even write down a complete strategy. It is unrealistic to think that we are close to techniques capable of solving this game. Though we cannot solve this game, we do have techniques that allow computer programs to play this game at a world-class level. Polaris, a computer poker player created by the University of Alberta's Computer Poker Research Group, beat world-class Texas Hold'em professionals by a statistically significant margin in the summer of 2008. This event was the first time a computer program has won a poker game (with statistical confidence) against human professionals.

Currently, the top computer poker programs use abstraction techniques to shrink these intractable games to a manageable size [2, 12, 7, 6, 8, 9, 22, 23]. These techniques are designed with the hope of maintaining the important strategic structure of the game while omitting the less relevant details.

These smaller abstract games are then solved with a state-of-the-art game-solver and the strategy found is played in the original game. The hope is that the abstract game captures enough of the structure of the original game so that the underlying strategy, which can play the abstract game perfectly, is in turn a good strategy in the original game.

This line of research is backed by the following underlying assumption:

Assumption 1 *By abstracting less, and therefore using richer strategy spaces and retaining more strategic detail of the original game, the quality of a strategy found by solving an abstract game should improve.*

This assumption has been stated multiple times (*e.g.*, [12, 7, 8, 23]), but no prior work has questioned its validity. Empirical evidence has shown that, in the domain of poker, solving larger abstract games has lead to stronger strategies. Roughly speaking, the AAAI Computer Poker Competition has been won every year by the team that solved the largest abstract game. Furthermore, the field of competitors has been stronger each year. This evidence of ever improving computer performance is apparent from the results of competitions against human players as well. In 2003, the first game-theoretic solutions to Texas Hold'em could compete with strong amateurs [2]. In just five years, the most advanced programs can now beat top-rated professional players. This empirical evidence, along with common sense, has lead us to believe that we are indeed more closely approximating solutions by increasing the size of our abstract games.

1.1 Contributions

This thesis makes the flowing contributions:

- **A formalization of abstraction and refinement along with a formal restatement of Assumption 1.**

Throughout this work, we will closely analyze Assumption 1. But before we can dissect this assumption, we must first formalize what it means to abstract an extensive game. We will provide a mathematical foundation defining what an abstraction is. With this formulation we can define a notion for when an abstraction is bigger or more informed than another. This notion, refinement, roughly means that the strategies available in a coarser abstraction are also available in a finer abstraction. A player using a finer abstraction has the same, and potentially more, information available to her at every point in the game where she must make a decision.

With this mathematical foundation in place, we will then formalize the underlying assumption that refining an abstract game will lead a better, less exploitable, strategy. From this, we arrive at multiple monotonicity assumptions, which formalize Assumption 1. Roughly speaking, a monotonic property holds if any refinement of an abstraction will lead to an improvement in terms of a strategy's exploitability.

- **Counterexamples to Assumption 1.**

We will disprove all of these assumptions by providing an abundance of counterexamples in variants of a small poker game, Leduc Hold'em. Furthermore, since the abstraction techniques we employ in these small poker games are similar to those used by top poker programs, these counterexamples lead us to believe that the monotonicity assumptions are unlikely to hold in the large poker games of interest. This work, along with the formalization of an abstraction, was published in *Abstraction Pathologies and Extensive Games* [21]. These two contributions are discussed in Chapter 3.

- **The introduction of domination value and its relationship to refinement.**

Despite our result that Assumption 1 does not hold, it is hard to throw all the empirical evidence aside completely. We continue our analysis by motivating why larger abstract games appear to produce higher quality strategies. We use the notion of a dominated strategy, which is a strategy that should never be played against any opponent, to argue that solutions to larger abstract games tend to make fewer mistakes. This is quite advantageous in settings like the AAAI Computer Poker Competition as the top poker programs typically do not attempt to exploit an opponent's weaknesses. That is, they benefit only from an opponent's errors, but make no effort to maximize this benefit. To measure the effect of playing dominated strategies, we introduce the domination value, an evaluation criteria that measures how much could be lost by a strategy making errors against a rational opponent. We then measure and compare the domination value of various Leduc Hold'em strategies with exploitability. We show that, though non-monotonicities still exist when using the domination value as our evaluation criterion, that strategies with lower domination values tend to do better in a tournament setting. That is, the domination value is highly correlated with what our intuition and empirical evidence has suggested in tournament settings. Furthermore, an arbitrary solution to an abstract game tends to be quite good compared to to the best possible solution in terms of its domination value, which is something we did not see under the exploitability analysis. This contribution is discussed in Chapter 4.

- **The introduction of a new algorithm, strategy grafting, which uses a divide and conquer approach to build solutions for larger abstract spaces than can be explicitly solved.**

With the insight that domination value could perhaps play a more important role than exploitability in regard to the quality of a strategy, we revisit the idea of using a divide and conquer approach, which can have a negative effect on exploitability, to allow for the use of larger strategy spaces. Decomposing an extensive game into sub-games, which are then solved separately, has been troublesome and error-prone when applied to games with imperfect information. In a game with imperfect information all the parts of a strong strategy work together. For example, in a poker game, if a player never bluffs then whenever she bets her observant opponent will know she has a good hand. That is, how she acts with the hands that she does not currently hold matters in terms of her

opponent's ability to determine which hand she actually does hold. Previous attempts to decompose games with imperfect information have failed to adequately address this issue. We propose a new method, called strategy grafting, which is designed to combat this problem. Here, we use sub-game decomposition, but each sub-game is solved with some additional shared information, the same base strategy. Though each sub-game is solved independently, the solution to a sub-game coordinates its play with the common base strategy. As a result, when the sub-games are combined they work well with each other in practice. We show, through theoretical analysis of this method, that we are almost guaranteed to improve the quality of our strategy if we know our opponent's abstraction. We also show empirically in both small and large poker games that this method does very well against opponents that use the current state-of-the-art methods even when their abstraction is not known. This contribution, which is discussed in Chapter 5, was published in *Strategy Grafting in Extensive Games* [20].

Chapter 2

Background

Before we describe our contributions, we must review some necessary background material. We begin by formalizing the extensive form game and the Nash equilibrium solution concept. After, we will review a variety of poker games as well as the abstraction techniques used by the state-of-the-art programs.

2.1 Extensive Games

Extensive games are a useful tool for modeling environments with multiple agents. Players and chance alternate taking actions until a terminal history is reached. At a terminal history, the game ends and each player receives a reward. However, players may not be able to completely observe the actual choices of the other players or chance. This allows for scenarios where two distinct sequences of actions may not be distinguishable by the player required to act, thus creating a game with imperfect information.

Definition 1 (Extensive Game) *[16, p. 200] A finite extensive game with imperfect information is denoted* Γ *and 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*.* $Z \subseteq H$ *is the set of* ***terminal histories****.* $A(h) = \{a : (h, a) \in H\}$ *are the actions available after a non-terminal history* $h \in H \setminus Z$*.*
- *A* ***player function*** P *that assigns to each non-terminal history 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*. Let* H_i *be the set of histories where player* i *chooses the next action.*
- *A function* f_c *that associates with every history* h *for which* $P(h) = c$ *a probability distribution* $f_c(\cdot|h)$ *on* $A(h)$*.* $f_c(a|h)$ *is the probability that* a *occurs given* h*.*

- *For each player $i \in N$, a partition $\mathcal{I}_i$ of H_i called the **information partition** of player i; a set $I_i \in \mathcal{I}_i$ is an **information set** for player i. For every $h, h' \in I_i$, we require that $A(h) = A(h')$. We shall denote I_h as the information set containing history h.*
- *For each player $i \in N$, a **utility function** u_i that assigns each terminal history a real value. $u_i(z)$ is rewarded to player i for reaching terminal history z. If $N = \{1, 2\}$ and for all z, $u_1(z) = -u_2(z)$, an extensive form game is said to be **zero-sum**.*

Histories in the same information set are indistinguishable to the player making the decision. This can result in some information partitions that force odd and unrealistic situations on a player where they are forced to forget their previous decisions. If all players can recall their previous actions and corresponding information sets, the game is said to be one of **perfect recall**. In this thesis, we will only consider games exhibiting perfect recall.

Matrix Games

One useful subclass[1] of extensive games are matrix games. In these games, all players choose their actions simultaneously and the combined joint-action defines the reward for each player. That is, each player has a single information set and chance never takes action. For example, Paper-Rock-Scissors (also known as Ro-Sham-Bo), is a zero-sum matrix game, which we display in Figure 2.1. Here, the values in the matrix are the rewards for the row-player for each joint action. For example, if the row-player plays *Rock* and the column-player plays *paper*, then the row-player will lose a reward of 1, and the column-player will gain a reward of 1. Matrix games are less interesting than extensive games, but we will use them later to illustrate some of our contributions.

	rock	paper	scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

Figure 2.1: An example matrix game: Paper-Rock-Scissors

2.1.1 Strategies and Solution Concepts

A **strategy for player i**, σ_i, in an extensive game, Γ, is a function that assigns a probability distribution over $A(h)$ to each $h \in H_i$, where $\sigma_i(h) = \sigma_i(h')$ for all h and h' in the same information set. That is, a strategy must give the same distribution over actions to all histories in the same information set. If player i is following σ_i, then whenever a history h is reached where $P(h) = i$, player i samples from $\sigma_i(h)$ to choose her action. We let Σ_i be the set of possible strategies for player i.

A **strategy profile**, σ, is a set consisting of a strategy for each player, $\{\sigma_1, \ldots, \sigma_n\}$, with σ_{-i} referring to the set of all the strategies in σ except σ_i.

[1] Matrix games and extensive games have the same expressive power, but extensive games are more succinct in many cases.

We define $u_i(\sigma)$ to be the expected reward for player i when all players play according to σ. For ease of notation, we let $u_i(\sigma_1, \sigma_2) = u_i(\{\sigma_1, \sigma_2\})$ and $u_i(\sigma_{-i}, \sigma_i') = u_i(\sigma_{-i} \cup \{\sigma_i'\})$.

Best Response

If player i knows the strategies of the other players, then she can compute a utility maximizing response. Given σ_{-i}, we say player i's **best response** is any strategy that maximizes

$$b_i(\sigma_{-i}) = \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_{-i}, \sigma_i'), \tag{2.1}$$

where $b_i(\sigma_{-i})$ is called the **best response value** to σ_{-i}.

Nash Equilibrium

Typically, the opposing players' strategies are not known in advance, so it is not possible for a player to simply compute a best response to determine her strategy. When a best response is unavailable, one often turns to the Nash equilibrium solution concept.

Definition 2 (Nash Equilibrium) *A **Nash equilibrium** is a strategy profile σ^* where for all players*

$$u_i(\sigma^*) = b_i(\sigma^*_{-i}) \tag{2.2}$$

*An approximation of a Nash equilibrium or **ε-Nash equilibrium** is a strategy profile σ where for all players*

$$u_i(\sigma) + \varepsilon \geq b_i(\sigma_{-i}) \tag{2.3}$$

If σ^* is a Nash equilibrium, then every player is playing a best response to σ^*_{-i} and therefore no player can benefit by deviating her strategy from σ^*. At an ε-equilibrium, no player can benefit more than ε by deviating from σ. All finite games have at least one Nash equilibrium. From here in, when we refer to an equilibrium we are referring to a Nash equilibrium.

2.2 Zero-sum Games

For the remainder of this work we will talk only of two-player zero-sum games with perfect recall. Adopting some conventions, we will refer to player 1 using female pronouns, and her opponent, player 2, using male pronouns. Unless specified otherwise, our goal will be to find a good strategy for player 1 to play against an unknown player 2.

Minimax Theorem

An **equilibrium strategy** is a strategy that belongs to some equilibrium in a zero-sum game. We denote the set of equilibrium strategies for player i as Σ_i^*. An equilibrium strategy maximizes a player's worst-case utility over all possible opponent strategies. This can also be thought of as minimizing the best-case utility of the player. Mathematically, this equivalence is stated in the Minimax Theorem [15].

Theorem 1 (Minimax Theorem) *For any zero-sum extensive game Γ, we have*

$$v^* = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_2 \in \Sigma_2} \max_{\sigma_1 \in \Sigma_1} u_1(\sigma_1, \sigma_2) \tag{2.4}$$

where v^* is the value of Γ for player 1.

As a corollary we have for all equilibrium σ^* and all $\sigma_1' \in \Sigma_1$, $\sigma_2' \in \Sigma_2$

$$v^* \quad = b_1(\sigma_2^*) \quad \leq u_1(\sigma_1^*, \sigma_2'), \text{ and} \tag{2.5}$$

$$-v^* \quad = b_2(\sigma_1^*) \quad \leq u_2(\sigma_2^*, \sigma_1') \tag{2.6}$$

Furthermore, for any σ we have

$$v^* \leq b_1(\sigma_2), \text{ and} \tag{2.7}$$

$$-v^* \leq b_2(\sigma_1) \tag{2.8}$$

In a zero-sum game we say it is *optimal* to play an equilibrium strategy because it guarantees the player the highest expected utility in the worst-case. Any deviation from equilibrium by the opponent can only benefit the player. Similarly, any deviation from equilibrium by the player can be exploited by a knowledgeable opponent. In this sense we can call computing an equilibrium in a zero-sum game *solving* the game.

Exploitability

An approximation of an equilibrium strategy can be evaluated by how far from the game's value an opponent can shift the game. We call this difference a strategy's exploitability.

Definition 3 *We define the **exploitability of player i** for a strategy σ_i, written $\varepsilon_i(\sigma_i)$, as*

$$\varepsilon_1(\sigma_1) = b_2(\sigma_1) + v^*, \text{ and} \tag{2.9}$$

$$\varepsilon_2(\sigma_2) = b_1(\sigma_2) - v^* \tag{2.10}$$

Exploitability measures how much an opponent who knows σ_i can benefit by player i's failure to play an equilibrium strategy. It is a worst-case bound on the quality of a strategy. This makes exploitability the natural metric for evaluating any technique whose goal is to compute a strong strategy with no knowledge of how an opponent might play. Note that for all σ_i, $\varepsilon_i(\sigma_i) \geq 0$ and for all equilibria σ^*, $\varepsilon_i(\sigma_i^*) = 0$. Also, note that the smallest ε for which a profile is an ε-equilibrium is $\max_i \varepsilon_i(\sigma_i)$.

2.2.1 Algorithms for Computing Equilibrium Strategies

In general, there is no known efficient algorithms for computing a Nash equilibrium. Fortunately, in the case of zero-sum games with perfect recall there are efficient techniques for computing an ε-Nash equilibrium. We will discuss these techniques in the next few sections.

Sequence Form

Sequence form was a huge breakthrough towards solving large extensive games. Prior to its existence, an extensive game was first converted into **normal form** (*i.e.*, a matrix game) and an equilibrium was then found using a linear program. Unfortunately, the size of a sequential game in normal form is typically exponential in the size of the game's description, which made its use impractical for all but toy games. Unlike normal form, the size of the **sequence form** representation of a game is linear in the size of the game's description. Furthermore, the sequence form description is itself a set of linear constraints, which allows us to employ linear programming techniques to solve for an equilibrium. The key insight that allows for all of this is that an arbitrary sequence of actions for a player, $\{(I_1, a_1), \ldots, (I_n, a_n)\}$, can be described uniquely by the last pair, (I_n, a_n) as long as the underlying game exhibits perfect recall.

The sequence form representation of a game is made up of the following components:

- An m by n **payoff matrix** A. We call the row player x and the column player y. Entry (i, j) of A is the reward for player x if x plays sequence i and y plays sequence j.
- A constraint matrix E and vector e. We say x is a valid sequence form strategy (called a **realization plan**) if $Ex = e$ and $x \geq 0$.
- A constraint matrix F and vector f. This pair is the analog of E and e for player y.

Every sequence of actions for a player, including the empty sequence, has an associated **realization weight**, which is a single entry in the realization plan. Let us denote the realization weight for sequence $\{(I_1, a_1), \ldots, (I_n, a_n), (I_{n+1}, a_{n+1})\}$ as $w(I_{n+1}, a_{n+1})$ and its parent sequence, $p(I_{n+1})$. The constraint matrices assign the realization weight for the empty sequence a value of 1. Furthermore, the sum of the realization weights of the form $w(I_{n+1}, *)$ must equal $w(p(I_{n+1}))$. An entry (i, j) in the payoff matrix, A, is assigned to be the expected utility for the row-player if the joint sequence of actions leads to a terminal history. If the joint sequence of actions does not end the game, or is invalid, the entry is set to 0.

Given a realization plan, one can compute the probability distribution at a history as:

$$\sigma(a|h) = \frac{w(I_h, a)}{\sum_{a' \in A(I_h)} w(I_h, a')} \tag{2.11}$$

Note that if the denominator is zero, then the history is never reached by the player and thus there is no need for us to compute the distribution. The consequence of this setup is, that given two realization plans, the expected utility for the row-player can be computed as yAx, which is bi-linear. This leads to simple linear programs for computing a best response.

Given a sequence form game and a strategy y, the best response for x and its value, $b_x(y)$, can be computed with the following linear program:

$$b_x(y) = \max_x yAx \quad \textbf{subject to} \quad Ex = e, \quad x \geq 0 \tag{2.12}$$

Though the best response linear program for y can be written similarly, it is useful to consider its dual program. That is, given a sequence form game and a strategy x, the best response value for y, $b_y(x)$, can be computed with the linear program:

$$b_y(x) = \min_u fu \quad \textbf{subject to} \quad Fu \leq Ax \tag{2.13}$$

From this, we can derive the linear program for computing a Nash equilibrium strategy for x as:

$$b_y(x^*) = \min_{u,x} fu \quad \textbf{subject to} \quad Fu \leq Ax, \quad Ex = e, \quad x \geq 0 \tag{2.14}$$

Koller and Pfeffer first made use of sequence form and linear programming in their GALA system to solve a variety of games [13]. The size of the games they solved with this technique are tiny compared to what can be solved today. This is mainly because linear programming packages have trouble maintaining the explicit sparsity of the payoff and constraint matrices. This can lead to a much higher memory requirement to actually solve an equilibrium than it does to write down the description of the linear program. For more details on sequence form, please refer to *Efficient Computation of Behavior Strategies* [19].

Excessive Gap Technique

The excessive gap technique is a non-smooth optimization technique that was devised by Nesterov [14]. It is an anytime first-order primal-dual algorithm that is guaranteed to be within ε of a solution after $O(\frac{1}{\varepsilon})$ iterations. Nesterov's algorithm is not applicable to all non-smooth optimization problems, hence how it surpasses the best-case convergence rate of $O(\frac{1}{\varepsilon^2})$ for general non-smooth optimization [14].

On each iteration, the excessive gap technique must compute a couple of matrix multiplications and solve a few simple optimization problems. The algorithm is quite fast in practice as these inner optimizations typically have a closed-form solution that can be computed rather quickly. Gilpin *et al.* showed that the excessive gap technique can be used along with the sequence form representation to solve for a Nash equilibrium in a zero-sum extensive game [6]. Since this technique maintains the sparsity of the payoff matrix, it can solve much larger games than a general linear program solver.

Counterfactual Regret Minimization

The concept of regret minimization is quite versatile. It is most easy to understand in the context of a multi-armed bandit problem. Here, at each time step the player must select a probability distribution over a fixed set of actions. After this selection, a bounded reward is assigned to each action (potentially by an all-knowing adversary) and the player receives her expected reward. The goal of the player is to minimize her **external regret**, which is the difference between the accumulated reward of the best single action and the total expected payoff she has received. That is, her regret is how much better off she would have been playing the best action for the entire time. If her regret

grows sublinearly as time passes, *i.e.*, her time-averaged regret approaches zero, we say that she has **no-regret**. For bandit problems, there exist many no-regret algorithms, such as Hedge [1] and regret-matching [10], which is based off Blackwell's approachability theorem [4].

The link between regret minimization and equilibria in zero-sum games is well-known. If two no-regret learners play a zero-sum game for T time steps and each has an average external regret of less than ε, then their average strategies form an 2ε-equilibrium.

PROOF. Let σ_i^t be the strategy used by player i on time step t, and σ_i^T be player i's time-averaged strategy. We are given:

$$\frac{1}{T} \max_{\sigma_i' \in \Sigma_i} \sum_{t=1}^{T} u_i(\sigma_i', \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t) \leq \varepsilon$$

Which by the linearity of expectation and the second term not depending on σ_i' gives

$$-\frac{\sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}^t)}{T} + \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}^T) \leq \varepsilon$$

Since $u_2(\cdot) = -u_1(\cdot)$, we can sum the two inequalities to get

$$\max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2^T) + \max_{\sigma_2' \in \Sigma_2} u_2(\sigma_1^T, \sigma_2') \leq 2\varepsilon$$

As $\forall x'$, $f(x') \leq \max_x f(x)$, for all i we have

$$u_{-i}(\sigma^T) + \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}^T) \leq 2\varepsilon$$

By again using $u_{-i}(\cdot) = -u_i(\cdot)$, we get our desired result of

$$b_i(\sigma_{-i}^T) \leq u_i(\sigma^T) + 2\varepsilon$$

■

Using this fact, one can apply Hedge or regret-matching directly to matrix games to solve for an equilibria, but they cannot be applied as-is to extensive games.

Recently, Zinkevich *et al.* showed how any regret minimizing algorithm could be efficiently applied in extensive games [23]. This required the introduction of the notion of **counterfactual utility** and **counterfactual regret**. The counterfactual utility for action a at information set I is the expected utility for taking action a given that information set I is reached weighted by the probability that chance and the opponent would play to reach I. If at each information set a no-regret learner minimizes counterfactual regret, which is its external regret with respect to counterfactual utility, then the external regret of the time-averaged strategy will also be minimized. Here, when we refer to the time-averaged strategy in an extensive game, we mean the average of the realization plan *not* the average of the action distributions at each information set.

This technique uses memory proportional to the resulting strategy profile and it is required to walk the entire game tree on each iteration. Zinkevich *et al.* also showed that a chance-sampled

variant of the technique had even better practical and theoretical performance for poker games. In this variant, the no-regret learners minimize an unbiased estimator of the counterfactual regret, as opposed to the actual counterfactual regret. The particular unbiased estimator allows one to sample chance's actions and, instead of walking the full game tree, merely play a single hand of poker on each iteration.

2.3 Poker Games

Poker games provide a versatile test-bed for many Artifical Intelligence techniques. First, they include imperfect information, unlike many other games that have been studied, such as Chess, Checkers and Go. Second, they can be easily scaled to a desired size and complexity by varying the number of cards in the deck, the number and size of the betting rounds, the number of betting options as well as the number of opponents. Third, many poker games are played by human professionals, which provides another method for evaluating the performance of our programs.

In this thesis we will use a variety of poker games to perform our experiments. Their rules are described in the next few sections.

2.3.1 Texas Hold'em

Texas Hold'em is a popular poker game that is played by people online, in casinos, and at home with friends for stakes from as low as pennies per hand up to as high as millions of dollars per hand. Because of its popularity, it has attracted the attention of artificial intelligence researchers, whose programs compete annually in the AAAI Computer Poker Competition [24].

Heads-up Texas Hold'em is a two-player poker game. One player is designated the *small blind* and the other the *big blind*. Each blind is required to make a forced bet into the pot. For the small blind this bet is one chip, for the big blind, this is two chips. A round of betting called the *preflop* occurs after each of the players has been dealt two private cards from a standard deck of shuffled cards. Three community cards are then dealt face-up for both players to see. A second round of betting called the *flop* occurs. Two subsequent rounds of betting called the *turn* and the *river* then occur and prior to each of these rounds a single community card is dealt face-up. After the river betting has finished and neither player has folded, the player with the best five card poker hand made from their private cards and the community cards wins all the chips in the pot.

The betting rounds all follow the same structure, regardless of the style of betting being used. The small blind acts first preflop. She may *call* or *bet*. Actions then alternate between the two players. When facing a bet, a player can *fold*, *call* or *raise*. Folding forfeits the chips in the pot to the opponent and the hand ends. Calling requires the player to match the bet faced and the betting round ends. Raising requires a player to match the current bet faced and place an additional bet into the pot. After a raise, the opposing player must then respond with an action of his own. If the first player to act initially checks or calls, then the next player has the option of checking, which ends

the betting round, or betting. The subsequent betting rounds are similar to the preflop, with two exceptions. First, the big blind is the first player to act in these rounds. This gives an advantage to the small bind, as she gets to see how the big blind acts prior to taking action herself. We say she has a positional advantage, or *position*. Second, the big blind can check as his initial action as he does not face a bet at the start of these rounds.

There are two betting variants that are commonly used when playing Texas Hold'em. The first is what is called **fixed-limit** betting. This is because there is a limit on the number of bets that can occur in a round and each bet is of a fixed size. On the preflop, there can be no more than three bets. All subsequent rounds allow for a maximum of four bets. On the preflop and flop, the bet size is fixed to be two chips, which is called a small bet. On the turn and river, the bet size doubles to four chips, which is called a big bet.

The second betting variant is called **no-limit** betting. With this style of betting, there are fewer restrictions on the amount of chips that a player may bet and there is no limit to the number of bets a player may make in a round. In no-limit, a player may bet almost any amount of his remaining chips. A bet size is valid if it is at least the size of the preceding bet on the same round or if the raise puts the player *all-in*, that is, the player bets all of her remaining chips. If no bet has been made, a player must bet at least two chips. In the no-limit Texas Hold'em game played in the AAAI Computer Poker Competition, each player starts with 400 chips.

Note that when we later refer to a poker game without mentioning the number of players, or style of betting, we mean the two-player fixed-limit variant.

2.3.2 Leduc Hold'em

Texas Hold'em, which has been the focus of much of the recent research, is too large to feasibly compute the exploitability of a strategy, much less an equilibrium profile. As a consequence, we chose to use a smaller poker game called Leduc Hold'em [18] for some of our experiments. In this game the deck contains two Kings, two Queens and two Jacks. Each player initially pays one chip into the pot, called an ante, and is dealt a single private card. After a round of betting, a community card is dealt face up. After a subsequent round of betting, if neither player has folded, both players reveal their private cards. If either player pairs their card with the community card they win the chips in the pot. Otherwise, the player with the highest private card is the winner. In the event both players have the same private card, they draw and split the chips in the pot. In Leduc Hold'em, the first player must act first on both betting rounds.

Like Texas Hold'em, fixed-limit Leduc Hold'em restricts the number and size of the bets that can be made. Here, the total number of bets in a round is not allowed to exceed two. Furthermore, each bet in the first round, the *preflop*, is fixed to be two chips. The subsequent betting round, the *flop*, has a fixed size of four chips. In no-limit Leduc Hold'em, each player starts with a thirteen chip stack (including the ante). Here, the minimum bet is one chip, as opposed to two in no-limit

Texas Hold'em.

2.4 Abstraction

Heads-up Texas Hold'em has approximately 10^{18} histories grouped into 10^{14} information sets. Because of its size, we cannot compute a Nash equilibrium for this game. One approach that has been successful in tackling this challenge is abstraction. Here, one models the large game with a much smaller game; one small enough that we can successfully compute an equilibrium. The smaller game, or **abstract game**, is created using an abstraction technique. The goal of an abstraction technique is to retain the important strategic features of the original game while reducing its size. We then use an equilibrium strategy found in the abstract game to play the original or **full game**. The hope here is that the equilibrium strategy from the abstract game will be a good approximation of an equilibrium strategy in the full game.

For large poker games, there are two types of abstraction that are commonly employed. We discuss these two types in the next two sections.

2.4.1 Card Abstraction

Card abstraction is typically done by grouping together hands of similar *strength* or distribution over future strengths. In the abstract game a player cannot distinguish between the hands in a group. The intuition is that hands of similar current or future strength can be played with a similar strategy. Grouping hands in this fashion requires a strength metric, typically placing some weight on the expected probability of winning at a showdown as well as on the variance of this probability. When using this method to abstract cards, the size of the abstract game is determined by the size of each hand group.

Johanson describes a method for creating abstractions using a metric called *hand strength squared* [12], which creates roughly equal sized groups based on a metric that incorporates both the expectation and variance of future strength. A method by Gilpin *et al.* uses a bottom-up approach to create a potential-aware abstraction [8]. This method, called *GameShrink*, works by clustering the terminal nodes and then repeatedly clustering earlier rounds based upon *distances* between distributions over future clusters. Both of these methods have proven to be effective in poker competitions.

2.4.2 Betting Abstraction

In the large no-limit games, using card abstraction alone is not enough to make manageable abstract games. Much of the size of the no-limit games comes from the large number of betting options available to the players. Here, we use betting abstraction, which typically restricts the number of betting options from each history. Each additional betting option has an exponential effect on the size of the game, so often only two or three bet sizes are made available to the player. A pot sized bet and an all-in bet are frequently among the ones chosen. When using the abstract strategy to play

in the full game, a program must *translate* the size of the bets in the game to and from actions in the abstract game. For example, if an opponent bets five chips when four chips were in the pot, the agent would likely translate this bet to a pot bet, or four chips. This translation can be particularly troublesome as an opponent can make awkward sized bets that do not map well to the abstract game. These bets distort the agent's belief of the number of chips in the pot and can lead to poor decision making.

Gilpin and colleagues describe the action abstraction and translation for their no-limit agent used in the 2007 AAAI Computer Poker Competition [9]. The main concept is that if the opponent makes a bet of size b that does not map directly into the abstract game, we compare that size to the surrounding bet sizes that do exist in the abstract game, for instance a and c, and translate it to the *closest* one. If $a < b < c$ then we define the closer size to be whichever ratio of $\frac{b}{a}$ or $\frac{c}{b}$ is smallest. This allows us to play an abstract strategy in the full game. Schnizlein *et al.* use a similar translation approach, but they allow for what is called *soft* translation [17]. Here, instead of using the ratios to translate to a single bet size, the ratios determine a probability distribution over which betting action was taken by the opponent. This usually reduces the exploitability of the strategy in the full game.

Chapter 3

Monotonicity and Abstraction Pathologies

3.1 Motivation

Extensive games have received considerable attention recently as the natural model for decision-making in poker. In fact, this research on artificial intelligence in poker has driven substantial algorithmic advancements in solving zero-sum extensive games. The traditional linear programming technique of Koller and Pfeffer [13] was unrivaled for over a decade, and could be used to solve games with 10^8 histories. In the past three years, a number of competing techniques have been developed by competitors in the AAAI Computer Poker Competitions [24], with some now able to solve games with about 10^{12} histories [23, 6].

While algorithmic advances have made it possible to solve very large extensive games, the smallest variants of poker played by humans are still much too large to solve. The standard approach to this problem is to use abstraction [2]. The idea is to construct (automatically or by hand) a tractably sized game that abstracts (ideally less important) details of the players' information at each decision. Hopefully, this smaller game retains much of the information of strategic structure. This game is constructed to be small enough that it is tractable to find a near equilibrium solution. One hopes this solution will hopefully perform well in the full game. The algorithmic advances in solving large extensive games allow for larger and larger abstractions. This improvement results in bigger abstract games, and presumably stronger strategies.

The dramatic advancements that have come out of the AAAI Poker Competitions, suggests this may be true. That is, bigger abstract games appear to result in stronger strategies. In the first year of the event, teams used linear programming techniques to solve small games, containing only a subset of the betting rounds [2, 7]. These small abstract games were then pieced together to create an overall poker strategy. In the following years, new equilibrium solving techniques were introduced and teams were able to solve a single abstract game to find a complete strategy [6, 22, 23]. In the following year, further enhancements and optimizations were made to solve even larger abstract games.

Each year, (roughly speaking) solving the largest abstract game has won the competition. Hence, a strategy of merely building and solving the largest possible (carefully constructed) abstracted games has been employed by the top teams, with the presumption that this will monotonically lead to stronger poker strategies. In fact, the success of this approach in the competition is evidence that the basic presumption may be true.

In this chapter, we begin by formalizing what it means to abstract a game. We then introduce a straightforward notion for what it means for an abstraction to be *bigger*. After this formalization, we examine the effect of abstraction on the strength of the resulting strategy in the worst-case, which is equivalent to saying how close of an approximation the strategy is to an equilibrium strategy. In particular, we examine the key presumption of the previous paragraph that finer abstractions will make the resulting strategy stronger in the full game. Our conclusion is that this presumption is not true in general. We, in fact, show a variety of *abstraction pathologies*: examples where refining an abstraction can result in a weaker strategy in the full game. These counterexamples are all in a small poker variant with only six cards, which allows us to compute exact measurements of a strategy's strength. The pathologies are also not limited to abstractions involving only chance nodes (common to all poker abstractions) or only players' actions (common to abstractions in no-limit variants).

3.2 Abstraction

Before we can quantify the effect that abstraction has on the quality of a strategy we must formalize the assumptions. This first requires us to define what it means to abstract a game. There are two methods for creating a smaller game from a larger one. First, we can merge information sets together. Second, we can restrict the actions a player can take from a history. We can also do a combination of these two methods.

Definition 4 (Abstraction) *An **abstraction for player i** is a pair $\alpha_i = \left\langle \alpha_i^{\mathcal{I}}, \alpha_i^A \right\rangle$, where,*

- *$\alpha_i^{\mathcal{I}}$ is a partitioning of H_i, defining a set of abstract information sets that must be coarser[1] than $\mathcal{I}_i$, and*
- *α_i^A is a function on histories where $\alpha_i^A(h) \subseteq A(h)$ and $\alpha_i^A(h) = \alpha_i^A(h')$ for all histories h and h' in the same abstract information set. We will call this the abstract action set.*

*The **null abstraction** for player i, is $\phi_i = \langle \mathcal{I}_i, A \rangle$. An **abstraction** α is a set of abstractions α_i, one for each player. Finally, for any abstraction α, the **abstract game**, Γ^α, is the extensive game obtained from Γ by replacing $\mathcal{I}_i$ with $\alpha_i^{\mathcal{I}}$ and $A(h)$ with $\alpha_i^A(h)$ when $P(h) = i$, for all i.*[2]

Strategies for abstract games are defined in the same manner as for unabstracted games, but the function must assign the same distribution to all histories in the abstraction information set, and

[1] Recall that partition A is coarser than partition B, if and only if every set in B is a subset of some set in A, or equivalently x and y are in the same set in A if x and y are in the same set in B.

[2] As discussed earlier some partitions of information can result in situations where a player is forced to forget their own past decisions. We only consider abstractions that result in an abstract game with perfect recall.

assign zero probability to actions not in the abstract action set. We will use the notation Σ_i^α to refer to the set of strategies for player i in the abstract game Γ^α.

3.2.1 Refinement

Having defined formally what it means to abstract a game, we must formalize what it means for one abstraction to be bigger or superior to another. We use the notion of refinement for this purpose.

Definition 5 (Refinement) *Let α_i and β_i be abstractions for player i. We say α_i **refines** β_i, written $\alpha_i \sqsupseteq \beta_i$, if and only if $\beta_i^\mathcal{I}$ is coarser than $\alpha_i^\mathcal{I}$ and $\alpha_i^A(h) \supseteq \beta_i^A(h)$ for all $h \in H_i$. If $\alpha_i \sqsupseteq \beta_i$ and $\beta_i \sqsupseteq \alpha_i$ then we say α_i and β_i are **equivalent**, written $\alpha_i \equiv \beta_i$. In addition, we say α refines β, or $\alpha \sqsupseteq \beta$, if and only if $\alpha_i \sqsupseteq \beta_i$ for all i.*

A refinement, therefore, allows the player's decisions to be contingent on all the same information (and possibly more). A refinement also allows a player to make all of the same choices (and possibly more). As a result, a refinement necessarily gives the player at least as many available strategies.

Theorem 2 *If $\alpha \sqsupseteq \beta$ then for all i, $\Sigma_i^\alpha \supseteq \Sigma_i^\beta$.*

PROOF. Suppose $\sigma_i \in \Sigma_i^\beta$. For any h and h' in the same abstract information set of α_i, they must also be in the same abstract information set of β_i since $\beta_i^\mathcal{I}$ is coarser than $\alpha_i^\mathcal{I}$. Therefore $\sigma_i(h) = \sigma_i(h')$. Furthermore, since $\sigma_i(h)$ defines a distribution over $\beta_i^A(h)$ it also defines a distribution over the superset $\alpha_i^A(h)$. Thus $\sigma_i \in \Sigma_i^\alpha$. ∎

3.3 Monotonicity

One might presume that by refining an abstraction, and thus having more available strategies, that better strategies would be found. However, solving games does not mean finding the best strategy from the available set, but rather finding a pair of strategies, one for each player, that are in equilibrium. The abstraction for each player matters. Still, one might presume that giving more available strategies to all players would result in a strategy that is closer to equilibrium in the full game. We can state these presumptions formally, using our notion of exploitability, $\varepsilon_i(\sigma_i)$, as the quantitative value for how far σ_i is from an equilibrium strategy.

Property 1 (Monotonicity) *Let α and β be abstractions, where $\alpha \sqsupseteq \beta$.*

(a) ***Strong Monotonicity for player i:*** *If σ^α and σ^β are Nash equilibria for Γ^α and Γ^β respectively, $\varepsilon_i(\sigma_i^\alpha) \leq \varepsilon_i(\sigma_i^\beta)$.*

(b) ***Weak Monotonicity for player i:*** *Let $\Sigma^{\alpha,*}$ and $\Sigma^{\beta,*}$ be the set of all Nash equilibria for Γ^α and Γ^β (respectively), $\min_{\sigma \in \Sigma^{\alpha,*}} \varepsilon_i(\sigma_i) \leq \min_{\sigma \in \Sigma^{\beta,*}} \varepsilon_i(\sigma_i)$.*

If the (strong or weak) monotonicity property for player i holds only if $\alpha_{-i} \equiv \beta_{-i}$, i.e.*, we only refine the player's abstraction, then we call it* ***player monotonic****. If it only holds if $\alpha_i \equiv \beta_i$,* i.e.*, we only refine the opponent's abstraction, then we call it* ***opponent monotonic****.*

Strong monotonicity requires that every equilibrium strategy in the finer abstract game be less exploitable than every equilibrium strategy in the coarser abstract game. This is the most hopeful property, and the one that is presumed by work that seeks to use larger and larger abstractions (since most game solvers make no claims about the equilibria they find or approximate). Extensive games, though, may have many equilibria, and some of these may be closer to optimal in the full game than others. Weak monotonicity states that the best equilibrium strategy in the finer abstract game is less exploitable than the best equilibrium strategy in the coarser game. Player and opponent (weak and strong) monotonicity are weaker monotonicity properties that refer to when a refinement only affects the player's or opponent's abstraction, respectively.

Unfortunately, none of these monotonicity properties hold in general. We show specific counterexamples of all of them in the counterexamples section. However, before moving on to these counterexamples, there are a few items we must discuss.

3.3.1 Null Opponent Monotonicity

First, there is one condition under which we can prove monotonicity does hold. If we use an abstraction where the opponent plays in the null abstraction (*i.e.*, the full game), then we see monotonicity in refining the player's abstraction.

Theorem 3 *Let α and β be abstractions where $\alpha \sqsupseteq \beta$ and $\alpha_2 \equiv \beta_2 \equiv \phi_2$. If σ^α and σ^β are Nash equilibria for Γ^α and Γ^β, respectively, then $\varepsilon_1(\sigma_1^\alpha) \leq \varepsilon_1(\sigma_1^\beta)$.*

PROOF. We use the definition of ε_1 (steps 3.1 and 3.4) and the Minimax Theorem (steps 3.2 and 3.4), plus the fact that a maximum over a set is no larger than the maximum over a superset (step 3.3) to prove this theorem.

$$
\begin{aligned}
v^* - \varepsilon_1(\sigma_1^\alpha) &= \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1^\alpha, \sigma_2) && (3.1)\\
&= \max_{\sigma_1 \in \Sigma_1^\alpha} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) && (3.2)\\
&\geq \max_{\sigma_1 \in \Sigma_1^\beta} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) && (3.3)\\
&= \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1^\beta, \sigma_2) = v^* - \varepsilon_1(\sigma_1^\beta). && (3.4)
\end{aligned}
$$

Removing v^* followed by negating the sides of the inequality and flipping the $\geq$ to $\leq$ gives us our result. ∎

Unfortunately, unless there is considerable asymmetry in the number of choices available to the two players, 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. Although the result gives some insight as to when abstraction is safe, it does not give a foundation for the practical use of abstraction.

3.3.2 Weak Monotonicity

Second, as alluded to earlier, not all abstract equilibrium strategies are equivalent in terms of exploitability in the full game. Strong monotonicity requires all equilibria in a finer abstraction to be stronger than all equilibria in a coarser abstraction. However, it may be that some equilibria are weaker or stronger than others. This is the premise of our weak monotonicity property. In this section we look into the best equilibrium for a particular abstraction, which first requires some new algorithmic developments.

In games that we can compute an equilibrium with an unabstracted opponent, we can efficiently find the best abstract equilibrium strategy using the following linear program:

$$\min_{u,v,x} gv \quad \textbf{subject to} \quad Gv \leq Bx, \quad Fu \leq Ax, \quad fu = t^*, \quad Ex = e, \quad x \geq 0 \tag{3.5}$$

Here (A, E, e, F, f) is the sequence form description of Γ^α and (B, E, e, G, g) is the sequence form description of a new game Γ^β. In Γ^β, $\beta_1 \equiv \alpha_1$ and $\beta_2 \equiv \phi$. t^* is the value of Γ^α.

Theorem 4 *Any x^* that is part of an optimal solution to (3.5) is an equilibrium strategy in Γ^α and no other equilibrium strategy of Γ^α is less exploitable in Γ.*

In our proof, we will refer to the sequence form linear programs found in background section 2.2.1.

PROOF. Let (u^*, v^*, x^*) be an optimal solution to (3.5). Clearly, (u^*, x^*) is a feasible point of (2.14) as the constraints of (2.14) are a subset of the constraints of (3.5). Furthermore, this point is optimal as (2.14) has optimal value t^* and the objective function, gu, is constrained in (3.5) to meet this condition. This implies x^* is an equilibrium strategy of Γ^α. Let x_0 be any equilibrium strategy of Γ^α. We can find u_0 and v_0 using (2.13) in Γ^α and Γ respectively. This implies (u_0, v_0, x_0) is a feasible point of (3.5). We note that if we only let v vary in (3.5) it is equivalent to (2.13) and therefore the value at (u_0, v_0, x_0) is $b_y(x_0)$. Finally, the value of (3.5) at a feasible point must be no less than the value at an optimal point. Therefore, $b_y(x^*) \leq b_y(x_0)$, which implies x_0 can be no less exploitable than x^*. We note that the best response values in this proof are in Γ^β, but since our opponent's abstraction is ϕ, these best response values are the same as in Γ. ∎

Unfortunately, solving (3.5) requires as much work as solving Γ^β, which involves a full representation of one player's strategy space. However, we can still use this approach to examine abstractions in small games, allowing us to explore the weak monotonicity property. Solving for the worst case equilibrium strategy, although interesting, does not appear to be tractable for even small games. Therefore, we do not examine it here.

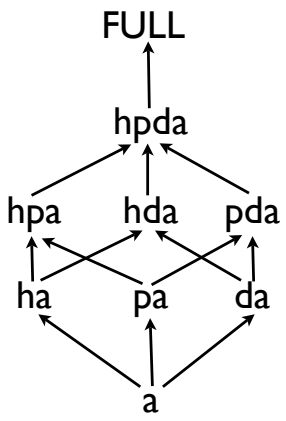

Figure 3.1: Visual representation of no-limit Leduc Hold'em action abstraction refinements

3.4 Counterexamples

To show counterexamples to all the useful monotonicity properties, we make use of Leduc and no-limit Leduc Hold'em. Both of which are described in the background, Section 2.3. These games are both small enough that abstraction is not necessary to solve for an equilibrium, but they are complex enough that we can see the effects that abstraction has on the exploitability of a strategy.

In our poker games, we use **millibets per hand** (or mb/h) to evaluate how well one program does against another or to describe how exploitable a program is. A millibet is one thousandth of a small bet in the fixed limit games or one thousandth of the smallest bet available in the no-limit games.

3.4.1 No-limit Leduc Hold'em

We will start by showing counterexamples to *action abstraction* in no-limit Leduc Hold'em. All the experiments in this section were performed by fellow Master's student David Schnizlein.

In the abstract games, each player has up to four betting options. These options are 50% pot (*h*), 100% pot (*p*), 200% pot (*d*) and all-in (*a*). Different abstractions are created by disallowing one or more of *h*, *p*, and *d*. When half pot or double pot are disallowed, any bets that would have been translated to them are instead translated to a pot bet. Similarly, if pot is also disallowed then all bets are translated to all-in. This ensures that an abstraction with more bets, like *hpda*, is a refinement of an abstraction with fewer bets, like *pda*. Figure 3.1 shows this visually. Note that this is not the translation described by Gilpin *et al.* as their technique may not *refine* an abstraction with the addition of new betting choices. The betting abstractions we use are *FULL*, *hpda*, *hpa*, *pda*, *pa*, and *a*. Though these betting abstractions were hand-picked, they are representative of abstractions selected by teams competing in the computer poker competitions.

Since we can abstract the player and the opponent separately, we denote an abstract game by the

Abstraction	Exploitability
FULL-FULL	0.0
hpda-FULL	0.4
pda-FULL	0.4
hpa-FULL	2.8
pa-FULL	2.8
a-FULL	5.7
hpda-hpda	94.0
FULL-hpda	103.2
a-hpda	116.8
pda-hpda	118.5
hpa-hpda	135.2
pa-hpda	138.3
hpa-hpa	185.4
pa-hpa	185.8
pda-hpa	190.4
FULL-hpa	193.7
a-a	238.0
pa-a	246.2
FULL-a	247.7
pda-pda	255.4
FULL-pda	276.1
pa-pa	373.3
FULL-pa	390.4

Table 3.1: Exploitability of various no-limit Leduc Hold'em strategies

player's abstraction followed by the opponent's abstraction. For example, the abstract game *hpda-pa* is the game where the player uses the *hpda* abstraction and the opponent uses the *pa* abstraction. For each abstract game, we found a single equilibrium strategy for the player using the sequence form linear program and CPLEX [11]. The exploitability of that strategy in the full game was then computed. A subset of the results are displayed in Table 3.1.

We see that when our opponent uses the *FULL* abstraction that the exploitability of our various abstractions are ordered correctly as required by Theorem 3. We also see that this ordering breaks as soon as our opponent is abstracted. One example is that *a-a* is only 238.0 mb/h exploitable while *pa-a* is 246.2 mb/h exploitable. Similarly, *pa-pa* is even more exploitable at 373.3 mb/h. Again, these are counterexamples to strong player and strong opponent monotonicity. In other words, giving the player or the opponent more actions can result in a more exploitable strategy. These examples show that none of the strong monotonicity properties hold when using action abstraction.

3.4.2 Leduc Hold'em

We used Leduc Hold'em to explore the effects of *card abstraction* on exploitability. Here, we explore both weak and strong monotonicity. We start by describing our card abstractions and their effects on strong monotonicity.

In our experiments, we start by abstracting the initial deal of the private card. For example,

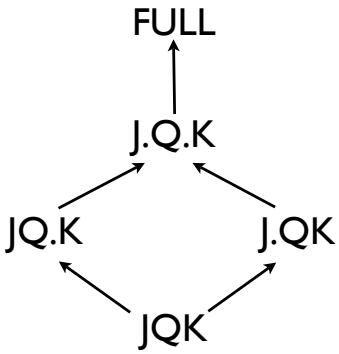

Figure 3.2: Visual representation of Leduc Hold'em card abstraction refinements

normally the player can separate their private card into King (*K*), Queen (*Q*), or Jack (*J*). One abstracted view is that the private card is either a King (*K*) or not a King (*JQ*). Similarly, an abstracted view of the community card is either the private card was paired (*pair*), or it was not (*nopair*). The abstractions we used in our experiment are *FULL*, *J.Q.K/pair.nopair*, *JQ.K/pair.nopair*, *J.QK/pair.nopair*, and *JQK/pair.nopair*. Here, *FULL* denotes the null abstraction, while periods between the cards separate information sets. For simplicity's sake, we drop *pair.nopair* from the description of the abstractions as it does not add ambiguity. For example, the *JQ.K* abstraction means that the player cannot differentiate between a Jack and a Queen preflop. Note that *J.Q.K* refines both *JQ.K* and *J.QK*, both of which refine *JQK*. Figure 3.2 shows these refinements visually. Though these card abstractions were hand-picked, they are similar to the abstractions that would be found by the automatic abstraction techniques that are used in the computer poker competitions.

A subset of the results of this experiment are displayed in Table 3.2. The abstraction in this table refers to the abstract game defined by abstracting both players as in the previous experiments. Like in the no-limit experiments, we solved a single equilibrium strategy for the player using the sequence form linear program and CPLEX [11] and computed its exploitability in the full game.

As is required by Theorem 3, monotonicity holds when our opponent uses the *FULL* abstraction. However, there are several examples where monotonicity fails to hold when our opponent is abstracted. For instance, *JQ.K-JQ.K* is only exploitable by 272.2 mb/h whereas *JQ.K-J.Q.K* is exploitable by 274.1 mb/h. In turn, this strategy is even less exploitable than the one for *J.Q.K-J.Q.K*, which is exploitable by 359.9 mb/h. These are explicit counterexamples to both strong player monotonicity and strong opponent monotonicity. In other words, refining the card abstractions for both players, as is classically done for competitions, can produce a more exploitable strategy.

Though we cannot tractably compute the best equilibrium strategies in many games we are interested in, we can examine them in Leduc Hold'em. In Table 3.3 we show a subset of the results of an experiment, which uses the same card abstractions from our first limit experiment. A new column is added to Table 3.2, which refers to the exploitability of the *best* equilibrium strategy. That

Abstraction	Exploitability
FULL-FULL	0.0
J.Q.K-FULL	55.2
JQ.K-FULL	69.0
J.QK-FULL	126.3
JQK-FULL	219.3
JQ.K-JQ.K	272.2
JQ.K-J.Q.K	274.1
FULL-J.QK	345.7
FULL-JQ.K	348.9
J.Q.K-J.Q.K	359.9
J.Q.K-JQ.K	401.3
J.QK-J.QK	440.6
FULL-JQK	459.5
FULL-J.Q.K	491.0
JQK-JQK	755.8

Table 3.2: Exploitability of various limit Leduc Hold'em strategies

is, the exploitability of the least exploitable abstract equilibrium strategy in the full game.

We find that even when we use the best equilibrium that we still see counterexamples to monotonicity. The exploitability of the best equilibrium strategy in the *J.Q.K-J.Q.K* abstract game is 358.6 mb/h, whereas the best equilibrium for *JQ.K-J.Q.K* is only exploitable by 78.8 mb/h, violating weak player monotonicity. Perhaps even more disturbing is the fact that *JQ.K-JQ.K*, which is obtained from the previously mentioned game by merging the Jack and Queen for both players, is only exploitable by 272.2 mb/h. In this case, even if we managed to converge to the best strategy in the larger game it is still worse than one we happened to converge to in the smaller game.

When ordered by the least exploitable equilibrium strategy we see many opposite trends as when ordered by the exploitability of the strategy that the linear program solver happened upon. First, if we could indeed find the best strategy, it appears we would like to give our player as much information about the game as possible. There exists an equilibrium strategy in the abstract game *FULL-J.Q.K* that is only exploitable by 1.7 mb/h, however our equilibrium solver happened upon one exploitable by 491 mb/h. This difference is incredible when you consider we can guarantee ourselves a strategy that is exploitable by 219.3 mb/h when using *JQK-FULL*, in which our player does not have any knowledge of her own private card prior to the community card's arrival! Second, even when giving our player as many options as possible, there is a counterexample to weak opponent monotonicity as we would rather give our opponent *JQK* than the refined *J.QK*. Finally, there is an example of both giving ourselves or our opponent more options leading to higher best case exploitability. Specifically, *JQ.K-J.Q.K* is better than *J.Q.K-J.Q.K* and *J.Q.K-JQ.K* is better than *J.Q.K-J.Q.K*.

Abstraction	**Exploitability**	**Exploitability** (Best)
FULL-FULL	0.0	0.0
FULL-J.Q.K	491.0	1.7
FULL-JQK	459.5	10.1
FULL-J.QK	345.7	45.3
J.Q.K-FULL	55.2	55.2
FULL-JQ.K	348.9	57.7
JQ.K-FULL	69.0	69.0
JQ.K-J.Q.K	274.1	78.8
J.Q.K-JQ.K	401.3	88.8
J.QK-FULL	126.3	126.3
JQK-FULL	219.3	219.3
JQ.K-JQ.K	272.2	272.2
J.Q.K-J.Q.K	359.9	358.6
J.QK-J.QK	440.6	440.6
JQK-JQK	755.8	710.2

Table 3.3: Best-case Exploitability of various limit Leduc Hold'em strategies

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

Figure 3.3: An example matrix game to demonstrate abstraction pathologies

3.5 Discussion

We used a smaller game, Leduc Hold'em, earlier so that we could analyze abstract game strategies in terms of their exploitability in the full game. However, we did not carefully construct this game or the abstractions to create the counterexamples we have presented. Leduc Hold'em is a scaled version of Texas Hold'em in which the number of cards was reduced and the betting simplified. The counterexamples arose naturally from the same kind of abstractions used in Texas Hold'em – card abstractions that group hands with similar *strength* and *potential* and restricting the betting actions. Since these pathologies are common in this small poker game, we expect them to also be common in larger poker games.

We now introduce an even simpler game to further illustrate how these pathologies arise. Consider the zero-sum matrix game shown in Figure 3.3. As mentioned in the background, matrix games are a special case of extensive games where players simultaneously choose an action. Their joint decision identifies an entry in the matrix specifying the utility for the *row-player*, where the utility for *column-player* is simply the negation of this entry. In this game, the row-player has two actions, A and B, and the column-player has four actions, a, b, c, and d. An abstraction of this game is obtained by eliminating one or more actions for either player.

Figure 3.4 gives a visual representation of this game. The x-axis shows the row-player's strategy space. For example, a point 30% along the x-axis represents a strategy where the row-player selects

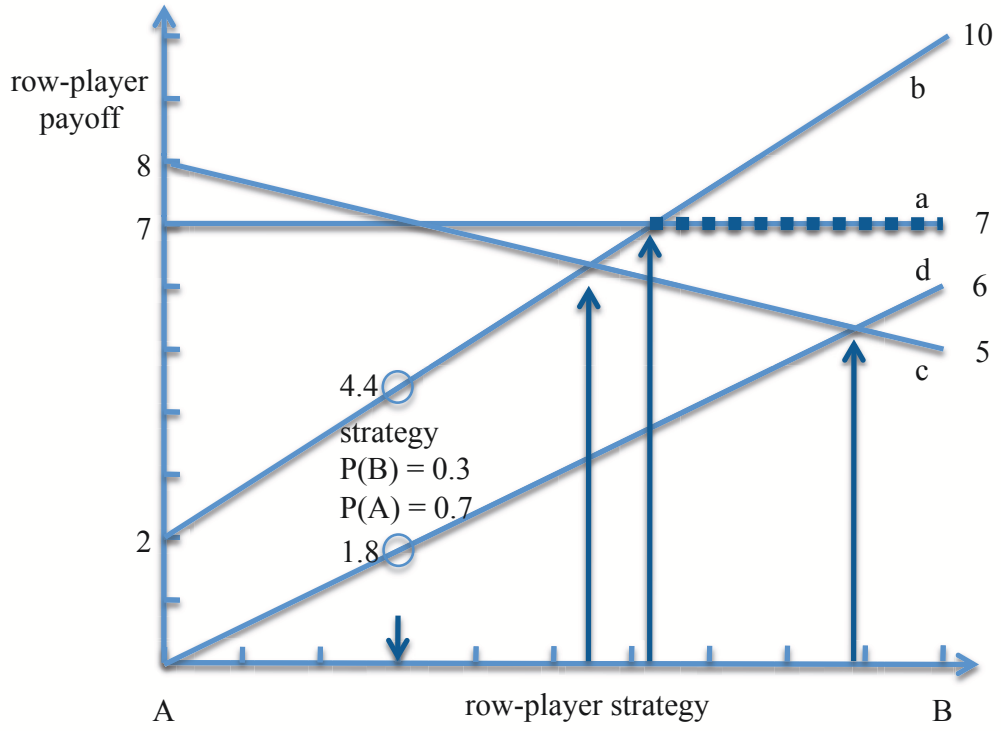

Figure 3.4: Visual representation of the matrix game in Figure 3.3

action *B* with probability 0.3 and action *A* the remainder of the time. Each line corresponds to a different action for the column player (*a*, *b*, *c* or *d*). The height of the line above any point on the x-axis represents the row-player's expected utility when the row-player plays that particular strategy and the column-player plays the action represented by the line. For example, if the row-player plays the strategy $P(B) = 0.3$ and the column-player selects action *b*, then the utility for the row-player is approximately 4.4, whereas if the column-player selects action *d* the utility for the row player is approximately 1.8. These points are circled in Figure 3.4. An equilibrium strategy guarantees the best worst-case value, which in the full game occurs at the intersection of lines *c* and *d*, giving at least a value of slightly more than 5 to the row player.

Consider an abstract game where the column-player only has actions *a* and *b*. This abstract game has a range of equilibrium strategies for the row-player corresponding to the dash-emphasized segment of line *a*. Not only are there many equilibrium strategies in this abstract game, but some of them are *better* than others in terms of their worst-case utility in the full game. Since the true equilibrium strategy for the row player lies within this space, this abstract game has a solution that is not exploitable. Now consider a refinement of this abstraction where the column-player has action *c* in addition to *a* and *b*. Here, the equilibrium strategy for the row-player shifts to the left to the intersection of lines *b* and *c*. The addition of this action removes the true equilibrium strategy from

the set of equilibrium strategies in the abstract game. More importantly, this unique equilibrium strategy has a lower worst-case utility in the full game than any equilibrium strategy in the coarser abstract game. Why does this occur? In essence, providing an additional action to an opponent has made the stronger strategies (including the true equilibrium) appear less attractive as the opponent lacks the ability to play strategies that would show that the alternatives are indeed worse.

Now, we show a pathology that arises from abstracting the row-player's actions. Suppose that the row player is only allowed to take action *B* and the column-player can only take action c. The equilibrium strategy is the only strategy for the row-player, $P(B) = 1$, and the row-player has a worst-case utility of 5. If we refine the game to allow the row player to also take action *A*, the equilibrium strategy moves to $P(A) = 1$, so the row player increases its utility to 8. However, in the full game this strategy has utility 0 for the row-player in the worst-case. Why does this occur? Providing additional strategies to a player can encourage the player to exploit the limitations of the opponent's abstraction, resulting in a strategy that is more exploitable by actions that become available to the opponent in the full game.

Chapter 4

Domination Value

4.1 Motivation

In the previous chapter, we showed that not only do abstraction pathologies exist but they appear to be common. This runs counter to competition experience and intuition that solving increasingly larger abstraction games causes monotonic improvement in performance. One potential reason that these pathologies have not proven problematic in competitions is that when we evaluate strategies in the tournament setting, most of the strong programs are playing equilibrium-like[1] strategies. That is, the strong programs are not adapting to their opponent's strategy to exploit any apparent weaknesses; they win primarily due to poor opponent play.

If we examine Table 4.1 and Table 4.2, we can see the results of the Leduc Hold'em strategies from the previous chapter played against each other in a tournament setting modeled after the AAAI Poker Competitions. Just as in these competitions, we analyze the outcome of the tournament in two fashions. First, Table 4.1 shows the results from what is known as a *bankroll* competition. Here, the strategies are ordered by their average expected winnings against all opponents. Second, Table 4.2 shows the results from what is known as a *runoff* competition. Here, the strategy with the lowest expected winnings is assigned last place. This strategy is then removed and the runoff competition is repeated using the remaining opponents until only the winner remains. In a runoff competition it is sufficient to to have positive expectation against every opponent, regardless of the amount, to win the competition. However, the winner of the bankroll competition can lose to some opponents as long as the difference is made up by beating the weaker opponents by a large amount. We should note that the strategies here play both positions against every opponent and the expected win rate is the average of the win rates over both positions. Similarly, the exploitability of a strategy is the average exploitability over both seats. To further clarify, the strategy *J.Q.K-JQ.K* is two ε-Nash equilibrium strategies, one for each position in Leduc Hold'em, both of which are solved with the *J.Q.K* abstraction for the player and the *JQ.K* abstraction for the opponent. A crosstable showing the individual matches, from which these tables are derived, is provided in the appendix.

[1] An equilibrium-like strategy is a static strategy that seeks to balance its play in order to prevent exploitation by the opponent. Strategies found by solving an abstract game are equilibrium-like.

There are a couple of interesting observations that we can gather from the tournament results. First, though the equilibrium does not lose to any other strategy and wins the runoff competition, it only places fourth in the bankroll competition. That is, it does not win as much against the weaker strategies as some of the other off-equilibrium strategies. Second, the exploitability of the strategies appears to only be weakly correlated with the tournament results with correlation coefficients of 0.15 and 0.30 for the bankroll and runoff tournaments respectively. If we look at the top 20% of strategies in terms of exploitability, which are all exploitable for less than 200 mb/h, the worst of them, *JQK-FULL*, finishes twenty-fourth in both tournaments. The remainder of these strategies are spread uniformly across the ranks. Third, the strategy that always folds is exploitable by 500 mb/h and would place last in either tournament format. Though most of the strategies are indeed less exploitable than always fold, *J.Q.K-JQK* is exploitable for over 600 mb/h and yet it places near the middle of the tournament in both settings. Finally, the largest strategy in which both players are abstracted, *J.Q.K-J.Q.K*, is the highest placed such strategy in the runoff competition, which is what would have been predicted from previous competition experience. The conclusions we can draw here are that, though exploitability says that refining an abstraction can lead to worse strategies, the tournament results still tend to follow our intuition that the larger abstract equilibrium strategies tend to perform better.

To help understand these results, let us investigate the well-known game of Paper-Rock-Scissors. The equilibrium strategy in this game is to play each action uniformly at random. The most exploitable strategy in this game would be to always play the same action (*e.g.*, always play rock). The equilibrium strategy will not only in expectation tie the most exploitable strategy, but it will in expectation tie *any* strategy. This guarantees that it cannot finish on top of a bankroll style tournament unless all but one of the other entrants to the competition is an equilibrium. That is, playing the equilibrium strategy in a Paper-Rock-Scissors tournament is a bad idea [3]. Despite this, equilibrium-like strategies do well in poker games. One possible explanation for this difference is the presence of dominated strategies in poker games, which do not exist in Paper-Rock-Scissors. We define a dominated strategy as follows:

Definition 6 (Dominated Strategy) *A **dominated strategy for player i** is a pure strategy[2], σ_i, such that there exists another strategy, σ_i', where for all opponent strategies σ_{-i},*

$$u_i(\sigma_i', \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i}) \tag{4.1}$$

*and the inequality must hold strictly for at least one opponent strategy. We say σ_i' **dominates** σ_i.*

This implies that a player can never benefit by playing a dominated strategy. It is, in essence, a mistake to play a dominated strategy against *any* opponent. Furthermore, an equilibrium strategy will never play a dominated strategy. Identifying dominated strategies is difficult in large games as it requires a linear program of similar size to one for solving the game [5].

[2] A pure strategy is one that assigns a probability of 1 to a single action at every history.

Rank	Strategy	Exploitability
1	FULL-J.QK	270.999
2	FULL-J.Q.K	388.18
3	FULL-JQ.K	334.159
4	FULL-FULL	0
5	J.Q.K-JQ.K	385.69
6	J.Q.K-J.QK	342.33
7	J.Q.K-J.Q.K	320.394
8	J.Q.K-FULL	53.8288
9	JQ.K-JQ.K	266.523
10	JQ.K-J.Q.K	228.787
11	JQ.K-J.QK	228.977
12	J.Q.K-JQK	620.34
13	FULL-JQK	543.635
14	JQ.K-FULL	84.8595
15	J.QK-J.QK	372.979
16	JQ.K-JQK	500.985
17	J.QK-JQ.K	378.093
18	J.QK-J.Q.K	256.532
19	J.QK-FULL	108.937
20	J.QK-JQK	563.338
21	JQK-JQ.K	239.901
22	JQK-J.QK	230.992
23	JQK-J.Q.K	242.141
24	JQK-FULL	199.876
25	JQK-JQK	610.737

Table 4.1: Bankroll Rankings and Exploitability for Leduc Hold'em Tournament

Rank	Strategy	Exploitability
1	FULL-FULL	0
2	FULL-J.QK	270.999
3	FULL-J.Q.K	388.18
4	FULL-JQ.K	334.159
5	J.Q.K-J.Q.K	320.394
6	J.Q.K-JQ.K	385.69
7	J.Q.K-FULL	53.8288
8	JQ.K-JQ.K	266.523
9	J.Q.K-J.QK	342.33
10	JQ.K-J.Q.K	228.787
11	JQ.K-J.QK	228.977
12	JQ.K-FULL	84.8595
13	J.QK-JQ.K	378.093
14	J.QK-J.Q.K	256.532
15	J.QK-J.QK	372.979
16	JQ.K-JQK	500.985
17	J.Q.K-JQK	620.34
18	FULL-JQK	543.635
19	J.QK-FULL	108.937
20	J.QK-JQK	563.338
21	JQK-JQ.K	239.901
22	JQK-J.QK	230.992
23	JQK-J.Q.K	242.141
24	JQK-FULL	199.876
25	JQK-JQK	610.737

Table 4.2: Runoff Rankings and Exploitability for Leduc Hold'em Tournament

	a	b
A	3	1
B	1	2
C	2	3

Figure 4.1: An example matrix game to demonstrate the effects of opponent refinement on dominated strategies

In this chapter, we will first explore some of the properties of domination and the effect of refinement on domination. Then, we will introduce a quantitative measure of how often a strategy makes mistakes, its domination value. Finally, we will use this measure to evaluate our Leduc Hold'em strategies from Chapter 3.

4.2 Properties of Domination and Refinement

As with exploitability, we can show a similar result for domination when we refine the player's abstraction against an unabstracted opponent.

Theorem 5 *If α and β are abstractions where $\alpha \sqsupseteq \beta$ and $\alpha_2 \equiv \beta_2$, then if σ_1 is a dominated strategy in Γ^β then σ_1 is also dominated in Γ^α.*

PROOF. Let $\sigma'_1 \in \Sigma_1^\beta$ be the strategy that dominates σ_1. Since $\Sigma^\beta \subseteq \Sigma^\alpha$, $\sigma'_1 \in \Sigma^\alpha$ by Theorem 2. Thus, Equation 4.1 holds in Γ^α as σ'_1 again dominates σ_1. ∎

Corollary 1 *If α is an abstraction and $\alpha_2 \equiv \phi_2$, then if σ_1 is dominated in Γ^α then σ_1 is dominated in Γ.*

The proof holds by direct application of Theorem 5. As a direct result of this corollary, if we refine the player's abstraction against an unabstracted opponent, we preserve strategy domination.

Theorem 5 states that refining the player's abstraction cannot add back in a strategy that was dominated in the coarser game. Unfortunately, the same is not true for refining the opponent's abstraction. For example, take the matrix game in Figure 4.1. In the abstract game where the row-player can play actions *A* and *B* and the column-player only *a*, we see that *A* dominates *B*. By refining the column player to also include action *b*, *B* is no longer dominated. Furthermore, *B* is dominated by *C* in the full game. That is, by refining the opponent we can reintroduce dominated strategies.

If we know that our opponent is rational, we know that they will not play any dominated strategies. This means that there could be additional strategies (that are not dominated) that we should never play since some strategies may become dominated once the opponent's irrational strategies are no longer considered. By both players assuming the other is rational, we get a notion known as iterated dominance.

Definition 7 (Iteratively Dominated Strategy) *An **iteratively dominated strategy for player i** is a pure strategy σ_i, such that either σ_i is a dominated strategy, or there exists a σ'_i such that for all*

	a	b
A	1	5
B	0	6
C	2	0
D	2	4

Figure 4.2: An example matrix game to demonstrate the effect of player refinement on iteratively dominated strategies

non-iteratively dominated opponent strategies σ_{-i} we have

$$u_i(\sigma_i', \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i}) \tag{4.2}$$

The inequality must hold strictly for at least one opponent strategy. Here, we say σ_i' ***iteratively dominates*** *σ_i.*

Note that the definition of iterative dominance is recursive. Again, like with a dominated strategy, it is a mistake to play an iteratively dominated strategy against an opponent who is and assumes rationality. As such, an iteratively dominated strategy will never be played by an equilibrium strategy.

Unlike with simple domination, iterative dominance is not necessarily preserved by refinement in any situation. It is straightforward to create examples of this using matrix games, as we previously did for opponent refinement. As an example, we see in Figure 4.2 an example of player refinement against an unabstracted opponent that leads to the re-introduction of an iteratively dominated strategy. In the abstract game where the only strategies available to the row-player are *A* and *B*, *a* dominates *b* (remember, the column-player is minimizing) and therefore *A* iteratively dominates *B*. By refining the player and introducing *C*, *a* no longer dominates *b*, so *B* is no longer iteratively dominated. In the full game, *D* dominates *C* and therefore *a* iteratively dominates *b*, which in turn makes *D* iteratively dominate both *A* and *B*. By refining the player in the first abstract game, we see that we can reintroduce a strategy that is iteratively dominated in both the coarser abstract game as well as the full game.

4.3 Domination Value

An equilibrium strategy makes an opponent indifferent to *all* non-iteratively dominated strategies. That is, to tie an equilibrium strategy in expectation, all one must do is play a non-iteratively dominated strategy. This is a much weaker condition than the requirement to play an equilibrium strategy that is necessary to be unexploitable. This observation leads us to a new evaluation criteria, called the domination value of a strategy.

Definition 8 (Domination Value) *The* ***domination value*** *of a strategy σ_i, written $\mathrm{Dom}_i(\sigma_i)$, is*

$$\begin{aligned} \mathrm{Dom}_1(\sigma_1) &= \max_{\sigma_2 \in \Sigma_2^*} u_2(\sigma_1, \sigma_2) + v^*, \text{ and} \\ \mathrm{Dom}_2(\sigma_2) &= \max_{\sigma_1 \in \Sigma_1^*} u_1(\sigma_1, \sigma_2) - v^* \end{aligned} \tag{4.3}$$

The domination value of a strategy measures how much value is lost against the worst-case equilibrium strategy. That is, by player 1's failure to assume her opponent is rational, how much additional value could she lose if her opponent is indeed rational. Note that for all σ_i, $\mathrm{Dom}_i(\sigma_i) \geq 0$ and for all equilibrium strategies σ_i^*, $\mathrm{Dom}_i(\sigma_i^*) = 0$. However, non-equilibrium strategies may also have a domination value of zero. In particular, any strategy that does not play any iteratively dominated strategies will have domination value zero.

As with exploitability, we have a series of monotonicity properties that we would like to hold under refinement using domination value instead of exploitability as our evaluation criteria. That is, our definitions for strong and weak monotonicity, as well as their respective player and opponent variations, from Chapter 3 remain unchanged after substituting Dom_i for ε_i. We should be rather pessimistic about any of these properties holding, especially after we have already shown that iterative dominance is not preserved under refinement. A second reason for this pessimism is that when we solve for an arbitrary equilibrium strategy in an abstract game, we have no control over how much weight is assigned to the iteratively dominated strategies, which is what will determine the domination value. That is, again there could be some equilibrium strategies from the same abstract game that are better or worse in terms of their domination values. Indeed, we will see in the results section that none of the monotonicity properties hold. We will, however, show that domination value is far more correlated with the outcome of a tournament than exploitability.

4.4 Computing the Domination Value of a Strategy

Before presenting our results, we must first outline how one computes the domination value of a strategy, as well as the domination value of the best equilibrium strategy in an abstract game. Like with exploitability, these computations are all done using the sequence form representation of a game and linear programming. Similarly, these linear programs are all about the size of the one used to solve an equilibrium in the full game and hence we can only compute the domination value of a strategy in small games.

We can compute the domination value of a strategy using the following linear program:

$$\mathrm{Dom}_2(y) = \max_{x,u} yA_1x - v^* \quad \textbf{subject to} \quad fu = v^*, \quad Fu \leq A_2x, \quad Ex = e, \quad x \geq 0 \tag{4.4}$$

Here, A_1 is the payoff matrix for the game where player 2 is abstracted and (A_2, E, e, F, f) is the sequence form representation of the full game.

PROOF. Let (x, u) be a feasible solution (4.4). (x, u) is a feasible point of (2.14), as the constraints of (2.14) are a subset of the constraints in (4.4). Additionally, (x, u) is in the solution set of (2.14) as the objective function, fu is constrained to be the optimal value of v^*. Therefore, any feasible x of (4.4) is an equilibrium strategy for player 1 in the full game. Let (x^*, u^*) be an optimal solution to (4.4). By definition, $u_1(x^*, y) = yA_1x^* \geq yA_1x = u_1(x, y)$ for all x satisfying the constraints

(or all equilibrium strategies for player 1). Therefore, x provides the greatest utility to player 1 of all equilibrium. ∎

As with the equilibrium linear program, it is useful for us to look at the dual linear program for computing the domination value of a strategy for player 1.

$$\mathrm{Dom}_1(x) = -\max_{a,b,c} fc - v^*a - v^* \quad \textbf{subject to} \quad Eb = ea, \quad Fc \leq A_2x + A_1b, \quad b \geq 0 \quad (4.5)$$

Again, A_1 is the payoff matrix for the game where player 1 is abstract and (A_2, E, e, F, f) is the sequence form representation of the full game. The dual variables have an interesting structure. In particular, b is a strategy for player 1, but this realization plan normalizes to a, instead of to 1. This means the expected payoff yAb is off by a constant factor of $\frac{1}{a}$ when a is non-zero. The vector c is similar to the dual vector in the standard best response linear program. In that program, the vector corresponds to the values of the opposing player's information sets.

As with the equilibrium linear program, we can modify the dual in a simple fashion to find player 1's best strategy in terms of domination value.

$$\max_{a,b,c,x,u} fc - v^*a \quad \textbf{subject to}$$
$$E_1b = e_1a, \quad F_1c \leq A_2x + A_1b, \quad f_3u = t^*, \quad F_3u \leq A_3x, \quad E_3x = e_3, \quad b, x \geq 0 \quad (4.6)$$

In this linear program, we make use of $(A_3, E_3, e_3, F_3, f_3)$, the sequence form representation of Γ^α, the game being solved by player 1, as well as t^*, the value of this game. In addition to the previous variables, we now let x vary. It is straightforward to show that (4.6) solves $\min_x \mathrm{Dom}_1(x)$, where x is constrained to be an equilibrium strategy, by substituting (4.5) for $\mathrm{Dom}_1(x)$.

4.5 Results

The first results we will present are the domination values for the Leduc Hold'em strategies from Chapter 3. Table 4.3 shows how each of these strategies does against the equilibrium in the full game found by CPLEX, against the worst-case equilibrium (the strategy's domination value) as well as the optimal domination value for an equilibrium strategy in the same abstract game. The results here are sorted by the optimal domination value, which we notice is typically close to the domination value of the arbitrary equilibrium. The only case where there is a difference of more than 20 mb/h is in the *FULL-JQK* abstraction. Similarly, the expected outcome against an arbitrary equilibrium is also typically close to the domination value of the strategy. Here, there is only two strategies with more than a 20 mb/h difference between the domination value and the expected loss to an arbitrary equilibrium. We also see that the larger abstract games tend to have lower domination values than the smaller abstract games. For example, many of the strategies with higher domination values use the *JQK* abstraction for one player.

Abstraction	EV of Equilibrium	Domination Value	Best Domination Value
FULL-FULL	0	0	0
J.Q.K-JQ.K	8.2628	9.6393	5.6056
FULL-JQ.K	15.9353	16.0543	9.2493
FULL-J.Q.K	13.7653	14.1158	14.1408
FULL-J.QK	23.1978	26.2033	15.7413
FULL-JQK	76.3623	83.9278	16.4323
JQ.K-JQ.K	24.4563	25.6953	25.5933
J.Q.K-J.Q.K	24.3738	27.8143	27.6663
JQ.K-J.Q.K	29.6123	32.5473	32.2413
J.Q.K-J.QK	38.2403	49.6428	38.8658
J.Q.K-FULL	31.0328	44.6188	44.5683
JQ.K-J.QK	39.9943	47.6378	47.6393
JQ.K-FULL	43.0438	54.1853	54.2448
JQ.K-JQK	67.3973	78.1593	58.7938
J.Q.K-JQK	89.3688	99.6298	74.3198
J.QK-JQ.K	71.6203	75.9163	75.4923
J.QK-JQK	89.1768	96.2983	75.7248
J.QK-FULL	99.3133	102.001	100.17
J.QK-J.Q.K	103.907	105.35	105.614
J.QK-J.QK	99.0023	121.061	121.346
JQK-J.Q.K	167.136	171.214	171.202
JQK-J.QK	172.795	175.228	175.737
JQK-JQ.K	177.037	179.729	179.612
JQK-FULL	186.56	189.777	189.602
JQK-JQK	351.151	402.385	390.788

Table 4.3: Domination Values for Leduc Hold'em Strategies

If we look more closely at the table, we can see that, as expected, none of the monotonicity properties hold. For example, *JQ.K-JQ.K* is slightly better than *J.Q.K-J.Q.K* in terms of best-case domination value for a violation of weak and strong monotonicity, *JQ.K-JQK* is better than *J.Q.K-JQK* for a violation of weak and strong player monotonicity and *J.QK-J.Q.K* is worse than *J.QK-JQ.K* for a violation of weak and strong opponent monotonicity. Though these violations exist, the magnitude of the violations is much less profound in comparison to the ones we found when using exploitability as our evaluation criteria.

Next, we look at how domination values of the strategies relate to the outcome of the tournament settings introduced earlier in this chapter. Table 4.4 and Table 4.5 show the bankroll and runoff rankings for the Leduc Hold'em tournaments with the domination values of the strategies instead of their exploitabilities. Here, if we look at the top 20% of strategies in terms of their domination value, which all have domination values less than 26 mb/h, we see a much higher concentration of strategies in the higher ranks. For instance, in the bankroll tournament, all of these top strategies are in the top seven places. Similarly in the runoff tournament, the top strategies are in the top six places. Though lower domination valued strategies tend to do better than higher ones in both tournaments, it is much more clear in the runoff tournament that lower domination valued strategies have an edge on higher

Rank	Strategy	Domination Value
1	FULL-J.QK	29.6861
2	FULL-J.Q.K	19.7281
3	FULL-JQ.K	25.8407
4	FULL-FULL	0
5	J.Q.K-JQ.K	13.4704
6	J.Q.K-J.QK	53.4826
7	J.Q.K-J.Q.K	25.3061
8	J.Q.K-FULL	38.6663
9	JQ.K-JQ.K	31.4793
10	JQ.K-J.Q.K	40.323
11	JQ.K-J.QK	53.1916
12	J.Q.K-JQK	84.4869
13	FULL-JQK	80.1915
14	JQ.K-FULL	53.8738
15	J.QK-J.QK	97.1951
16	JQ.K-JQK	69.592
17	J.QK-JQ.K	55.3213
18	J.QK-J.Q.K	78.0052
19	J.QK-FULL	87.131
20	J.QK-JQK	93.0526
21	JQK-JQ.K	149.648
22	JQK-J.QK	144.337
23	JQK-J.Q.K	144.059
24	JQK-FULL	152.766
25	JQK-JQK	284.756

Table 4.4: Bankroll Rankings and Domination Value for Leduc Hold'em Tournament

ones. For both types of tournaments we can see a strong correlation between domination value and the tournament rankings. The correlation coefficients here are 0.84 and 0.87 for the bankroll and runoff tournaments respectively. Here, we are referring to the correlation between tournament rank and domination value.

Rank	Strategy	Domination Value
1	FULL-FULL	0
2	FULL-J.QK	29.6861
3	FULL-J.Q.K	19.7281
4	FULL-JQ.K	25.8407
5	J.Q.K-J.Q.K	25.3061
6	J.Q.K-JQ.K	13.4704
7	J.Q.K-FULL	38.6663
8	JQ.K-JQ.K	31.4793
9	J.Q.K-J.QK	53.4826
10	JQ.K-J.Q.K	40.323
11	JQ.K-J.QK	53.1916
12	JQ.K-FULL	53.8738
13	J.QK-JQ.K	55.3213
14	J.QK-J.Q.K	78.0052
15	J.QK-J.QK	97.1951
16	JQ.K-JQK	69.592
17	J.Q.K-JQK	84.4869
18	FULL-JQK	80.1915
19	J.QK-FULL	87.131
20	J.QK-JQK	93.0526
21	JQK-JQ.K	149.648
22	JQK-J.QK	144.337
23	JQK-J.Q.K	144.059
24	JQK-FULL	152.766
25	JQK-JQK	284.756

Table 4.5: Runoff Rankings and Domination Value for Leduc Hold'em Tournament

Chapter 5

Strategy Grafting

5.1 Motivation

In the last chapter, we provided compelling evidence that using larger strategy spaces typically improves performance in a tournament setting against equilibrium-like opponents. Unfortunately, state-of-the-art methods for solving extensive games require enough system memory for at least one strategy profile. That is, the size of the largest solvable game will likely increase mainly due to hardware improvements. For this reason, if we wish to use larger strategy spaces we will have to devise new methods for computing strategies.

Decomposition is a natural approach towards increasing the expressive power of strategies and indeed it has previously been employed toward this end. Prior approaches that have employed decomposition in the AAAI Computer Poker Competition, such as solving betting rounds independently, have since been abandoned in favor of solving a single abstract game. This is because in extensive games with imperfect information decomposition can be problematic. One way that equilibrium strategies guard against strong opponents is through information hiding. The strategy plays in a fashion such that an observant opponent cannot effectively reconstruct and take advantage of the player's private information. Independent solutions to a set of sub-games, though, may not "mesh" well together. For example, an observant opponent might be able to determine which sub-game is being played, which itself could be valuable information that could be exploited. In this chapter, we introduce a new technique, strategy grafting, which attempts to alleviate this concern while using decomposition to afford us much larger strategy spaces.

5.2 Strategy Grafting

We will start by describing the strategy grafting algorithm and providing some theoretical results regarding the quality of grafted strategies. Unlike prior techniques, strategy grafting attempts to solve the sub-games in a fashion so that when they are combined they mesh well with each other, which should lead to good overall strategy.

5.2.1 Grafting Partition

In order to formally define a grafted strategy, we must first explain how a game can be divided into sub-games. We make use of a grafting partition for this.

Definition 9 (Grafting Partition) *Let $G = \{G_0, G_1, \ldots, G_p\}$ be a **grafting partition for player i** if and only if:*

1. *G is a partition of H_i,*
2. *$\forall I \in \mathcal{I}_i \; \exists j \in \{0, \ldots, p\}$ such that $I \subseteq G_j$, and*
3. *$\forall j \in \{1, \ldots, p\}$ if h is a prefix of $h' \in H_i$ and $h \in G_j$ then $h' \in G_j \cup G_0$.*

*G_0 is a special member of the partition. We call it the **base set**.*

In words, a grafting partition is a partition of the player's histories into sub-games. There are two restrictions on this partition. First, it cannot break up information sets across blocks of the partition. Second, there is no way to play out the original game such that histories from more than one non-base set are visited. That is, no sequence of actions can touch more than one non-base set. The second restriction is what allows for us to solve each graft independently without them interfering with each other.

5.2.2 Grafted Strategy

Using the elements of a grafting partition, we can construct a set of sub-games. The solutions to these sub-games are called grafts, and we can combine them naturally, since they are disjoint, into one single grafted strategy.

Definition 10 (Grafted Strategy) *Given a strategy $\sigma_i \in \Sigma_i$ and a grafting partition G for player i. For $j \in \{1, \ldots, p\}$, define $\Gamma^{\sigma_i, j}$ to be an extensive game derived from the original game Γ where for all $h \in H_i \setminus G_j$, $P(h) = c$ and $f_c(a|h) = \sigma_i(a|h)$. In other words, player i only controls her actions for histories in G_j, and is forced to play according to σ_i at all other decision points. Let the **graft** of G_j, $\sigma^{*,j}$, be an ϵ-Nash equilibrium strategy of the game $\Gamma^{\sigma_i, j}$. Finally, define the **grafted strategy for player i** σ_i^* as,*

$$\sigma_i^*(a|h) = \begin{cases} \sigma_i(a|h) & \text{if } h \in G_0 \\ \sigma_i^{*,j}(a|h) & \text{if } h \in G_j \end{cases}$$

*We will call σ_i the **base strategy** and G the grafting partition for the grafted strategy σ_i^*.*

There are a few key ideas to observe about grafted strategies that distinguish them from previous sub-game decomposition methods. First, we start out with a base strategy for the entire game. This base strategy can be constructed using current techniques for a tractably sized abstraction. It is important that we use the same base strategy for all grafts, as it is the only information that is passed

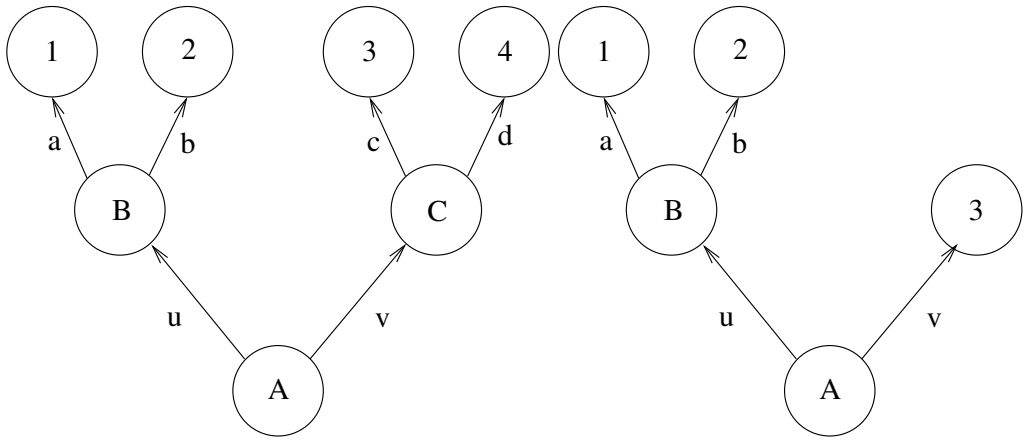

Figure 5.1: An example of Strategy Grafting

between the grafts. Second, when we construct a graft, only the portion of the game that the graft plays is allowed to vary for our player of interest. The actions over the remainder of the game are played according to the base strategy. This allows us to choose the size of each set in the grafting partition, so that it itself is as large as the largest tractably solvable game. Third, note that when we construct a graft, we continue to use an equilibrium finding technique, but we are not interested in the pair of strategies. Instead, we are only interested in the strategy for the player of interest over the grafting block of interest. This means in games like poker, where we are interested in a strategy for both players, we must construct a grafted strategy separately for each player. Finally, when we construct a graft, our opponent must learn a strategy for the entire (potentially abstracted) game. By letting our opponent's strategy vary completely, our graft will be a strategy that is less prone to exploitation and force each individual graft to mesh well with the base strategy. The hope is that if each graft meshes well with the base strategy that they will in turn mesh well with the other grafts when combined.

In Figure 5.1, we see an example of how strategy grafting works in a simple single player game with no elements of chance. Here, we use a grafting partition where $G_0 = \{C\}$ and $G_1 = \{A, B\}$. When solving for the graft, when play reaches history C she is forced to follow the base strategy to decide her action. In this case, we use a base strategy that always takes action c from history C, which results in the player receiving a reward of 3 for taking action v from history A. This altered game, which is displayed in the right portion of the figure, is what is solved to determine the grafted strategy.

5.2.3 Grafting Improvement Theorem

Strategy grafting allows us to construct a strategy with more expressive power that what can be computed by solving a single game. We now show that strategy grafting uses this expressive power to its advantage, causing an (approximate) improvement over its base strategy. Note that we cannot

guarantee a strict improvement as the base strategy may already be an optimal strategy.

Theorem 6 *For strategies σ_1, σ_2 where σ_2 is an ϵ-best response to σ_1, if σ_1^* is the grafted strategy for player 1 where σ_1 is used as the base strategy and G is the grafting partition then,*

$$u_1(\sigma_1^*, \sigma_2) - u_1(\sigma_1, \sigma_2) = \sum_{j=1}^{p} \Big(u_1(\sigma_1^{*,j}, \sigma_2) - u_1(\sigma_1, \sigma_2) \Big) \geq -3p\epsilon.$$

In other words, the grafted strategy's improvement against σ_2 is equal to the sum of the gains of the individual grafts against σ_2 and this gain is no less than $-3p\epsilon$.

PROOF. Define Z_j as follows,

$$\forall j \in \{1, \ldots, p\} \quad Z_j = \{z \in Z \mid \exists h \in G_j \text{ with } h \text{ a prefix of } z\} \tag{5.1}$$

$$Z_0 = Z \setminus \bigcup_{j=1}^{p} Z_j \tag{5.2}$$

By condition (3) of Definition 9, $Z_{j=0,\ldots,p}$ are disjoint and therefore form a partition of Z.

$$\sum_{j=1}^{p} \Big(u_1(\sigma_1^{*,j}, \sigma_2) - u_1(\sigma_1, \sigma_2) \Big) \tag{5.3}$$

$$= \sum_{j=1}^{p} \left(\sum_{z \in Z} u_1(z) \Pr(z|\sigma_1^{*,j}, \sigma_2) - \sum_{z \in Z} u_1(z) \Pr(z|\sigma_1, \sigma_2) \right) \tag{5.4}$$

$$= \sum_{j=1}^{p} \sum_{k=0}^{p} \sum_{z \in Z_k} u_1(z) \Big(\Pr(z|\sigma_1^{*,j}, \sigma_2) - \Pr(z|\sigma_1, \sigma_2) \Big) \tag{5.5}$$

Notice that for all $z \in Z_{k \neq j}$, $\Pr(z|\sigma_1^{*,j}, \sigma_2) = \Pr(z|\sigma_1, \sigma_2)$, so only when $k = j$ is the summand non-zero.

$$= \sum_{j=1}^{p} \sum_{z \in Z_j} u_1(z) \Big(\Pr(z|\sigma_1^{*,j}, \sigma_2) - \Pr(z|\sigma_1, \sigma_2) \Big) \tag{5.6}$$

$$= \sum_{j=1}^{p} \sum_{z \in Z_j} u_1(z) \Big(\Pr(z|\sigma_1^{*}, \sigma_2) - \Pr(z|\sigma_1, \sigma_2) \Big) \tag{5.7}$$

$$= \sum_{z \in Z} u_1(z) \Big(\Pr(z|\sigma_1^{*}, \sigma_2) - \Pr(z|\sigma_1, \sigma_2) \Big) \tag{5.8}$$

$$= \left(\sum_{z \in Z} u_1(z) \Pr(z|\sigma_1^{*}, \sigma_2) - \sum_{z \in Z} u_1(z) \Pr(z|\sigma_1, \sigma_2) \right) \tag{5.9}$$

$$= u_1(\sigma_1^*, \sigma_2) - u_1(\sigma_1, \sigma_2) \tag{5.10}$$

Furthermore, since $\sigma_1^{*,j}$ and $\sigma_2^{*,j}$ are strategies of the ϵ-Nash equilibrium $\sigma^{*,j}$,

$$u_1(\sigma_1^{*,j}, \sigma_2) + \epsilon \geq u_1(\sigma_1^{*,j}, \sigma_2^{*,j}) \geq u_1(\sigma_1, \sigma_2^{*,j}) - \epsilon \tag{5.11}$$

Moreover, because σ_2 is an ϵ-best response to σ_1,

$$u_1(\sigma_1, \sigma_2^{*,j}) \geq u_1(\sigma_1, \sigma_2) - \epsilon \tag{5.12}$$

So, $\sum_{j=1}^{p} \Big(u_1(\sigma_1^{*,j}, \sigma_2) - u_1(\sigma_1, \sigma_2) \Big) \geq -3p\epsilon$. ∎

This theorem leads directly to an important corollary.

Corollary 2 *Let α and β be abstractions, where $\alpha \sqsupseteq \beta$ and $\alpha_{-i} \equiv \beta_{-i}$. Let σ be an ϵ-Nash equilibrium strategy for the game Γ^{β}, then any grafted strategy σ_1^* in Γ^{α} with σ_1 used as the base strategy will be at most $3p\epsilon$ worse than σ_1 against σ_2.*

PROOF. By the definition of ϵ-Nash equilibrium, σ_2 is an ϵ-best response to σ_1 in Γ^{β}. Since the opponent's abstraction does not change, and $\sigma_1 \in \Sigma_1^{\alpha}$, σ_2 is also an ϵ-best response in Γ^{α}. Therefore the conditions of Theorem 6 hold and our result follows immediately. ∎

Although these results suggest that a grafted strategy will (approximately) improve on its base strategy against an equilibrium opponent, there is one caveat: it assumes we know the opponent's abstraction or can solve a game with the opponent unabstracted. Without this knowledge or ability, this guarantee does not hold. However, empirically we know that larger strategy spaces appear to perform better in tournament settings, so it would not be unreasonable to expect a similar result by using strategy grafting.

5.3 Results

5.3.1 Leduc Hold'em

First, we will train a variety of grafted strategies in Leduc Hold'em. We will analyze these strategies in the two tournament settings described in Chapter 4 as well as evaluate them in terms of exploitability and domination value. The abstractions we will use are the same as the Leduc Hold'em card abstractions from Chapter 3. As a brief reminder, all abstractions can only distinguish if they have paired or not postflop. Preflop, they may or may not group cards together.

We chose to train two types of grafted strategies: *preflop grafts* and *flop grafts*. Both types consist of three individual grafts for each player: one to play each card with complete information. That is, each graft does not abstract the observable cards. These two types differ in that the preflop grafts play for the entire game whereas the flop grafts only play the game after the flop. For preflop grafts, this means G_0 is empty, *i.e.*, the final grafted strategy is always using the probabilities from some graft and never the base strategy. For flop grafts, the grafted strategy follows the base strategy in all preflop information sets. We will use ε-Nash equilibria in the three abstract games as our base strategies. Each graft is trained with the opponent in the same abstraction as the corresponding base strategy. Contrary to the previous chapters, each base strategy and graft is trained using chance-sampled counterfactual regret minimization for one billion iterations, as opposed to a linear program. The equilibria found are ε-Nash equilibria where no player can benefit more than $\varepsilon = 10^{-5}$ chips by deviating within the abstract game.

In Table 5.1 and Table 5.2, we see the results of the Leduc Hold'em tournaments from Chapter 4 with the addition of our six grafted strategies. We see that the grafted strategies fair quite well in

Bankroll Rank	Strategy
1	FULL-J.QK
2	FULL-J.Q.K
3	FULL-JQ.K
4	FULL-FULL
5	J.Q.K-J.Q.K preflop grafts
6	J.Q.K-JQ.K
7	J.Q.K-J.Q.K flop grafts
8	J.QK-J.QK preflop grafts
9	J.Q.K-J.QK
10	JQ.K-JQ.K flop grafts
11	JQK-JQK preflop grafts
12	JQ.K-JQ.K preflop grafts
13	J.Q.K-J.Q.K
14	J.Q.K-FULL
15	J.QK-J.QK flop grafts
16	JQ.K-JQ.K
17	JQ.K-J.Q.K
18	JQ.K-J.QK
19	JQ.K-FULL
20	JQK-JQK flop grafts
21	J.QK-J.QK
22	FULL-JQK
23	J.Q.K-JQK
24	J.QK-JQ.K
25	JQ.K-JQK
26	J.QK-J.Q.K
27	J.QK-FULL
28	J.QK-JQK
29	JQK-JQ.K
30	JQK-FULL
31	JQK-JQK
32	JQK-J.QK

Table 5.1: Bankroll Ranking for Leduc Hold'em Tournament with Grafted Strategies

Bankroll Rank	Runoff Rank	Strategy	Exploitability	$\mathrm{Dom}(\sigma)$
5	4	J.Q.K-J.Q.K preflop grafts	486.067	22.5416
7	6	J.Q.K-J.Q.K flop grafts	532.631	25.1121
8	7	J.QK-J.QK preflop grafts	776.2	91.1531
10	11	JQ.K-JQ.K flop grafts	558.157	49.939
11	18	JQK-JQK preflop grafts	761.014	66.0652
12	8	JQ.K-JQ.K preflop grafts	639.748	37.0566
15	13	J.QK-J.QK flop grafts	621.605	96.4149
20	23	JQK-JQK flop grafts	780.344	118.394
		J.Q.K-J.Q.K	514.556	35.1924
		JQ.K-JQ.K	404.357	40.8234
		J.QK-J.QK	610.489	135.223
		JQK-JQK	986.812	343.825

Table 5.2: Runoff Ranking, Exploitability and Domination Value of Grafted Strategies

this field of equilibrium-like strategies. In particular, all of the grafted strategies, with the exception of *JQK-JQK flop grafts*, finish in the top half of the bankroll competition. Similar results are seen in the runoff competition, where only *JQK-JQK preflop grafts* and its flop graft counterpart are in the bottom half of the field. The base strategy seems to be of great importance when training a grafted strategy. Aside from *JQK-JQK preflop grafts* performing better than two of the two bucket grafts in the bankroll competition, the order of the grafts corresponds with the size of the base strategy space. Although the choice of base strategy is important, the grafted strategies do well under both evaluation criteria and even the worst base strategy sees great relative improvement when used to train grafted strategies. Another trend we see, is that the preflop grafts do better than the flop experts in all but one case. We notice that the *JQ.K-JQ.K flop grafts* strategy beats its preflop counterpart in the bankroll competition, but only by about 4 mb/h. In the runoff competition, all preflop grafts beat their corresponding flop grafts.

We see no concrete effect in terms of how the exploitability of a grafted strategy relates to that of its corresponding base strategy. For example, the *J.Q.K-J.Q.K preflop grafts* is about 30 mb/h better than its base strategy, but the *J.Q.K-J.Q.K flop grafts* is about 20 mb/h worse. All of the *JQ.K-JQ.K* and *J.QK-J.QK* grafts increase exploitability. Though from Chapter 4, we know that exploitability is not a criteria we can use to accurately predict a strategy's performance in these tournaments. In terms of the domination values of the grafts, we see improvement in all but one case. That is, the *JQ.K-JQ.K flop grafts* is 9 mb/h worse than its base strategy. As the domination value seems to be a better predictor of of how well a strategy will do in a tournament, it is not surprising that the grafted strategies typically improve in this regard. That is, the grafted strategies tend to make fewer mistakes, either due to them having fewer dominated strategies, or placing less weight on those dominated strategies.

5.3.2 Texas Hold'em

We used strategy grafting to create a heads-up Texas Hold'em poker program. We used our entrant to the 2nd Man vs. Machine Poker Challenge, which occurred in Las Vegas in July 2008, as the base strategy and created a grafted strategy approximately 12.2 times larger using similar abstraction techniques. This program was entered into the 2009 AAAI Computer Poker Competition. It finished second in the runoff competition, and third in the bankroll competition out of a field of thirteen competitors.

In Table 5.3 we see the results of running this large grafted strategy against a variety of our equilibrium-like strategies. The *20x32* strategy is the largest single abstract game we can solve to date. It is approximately 2.53 times larger than the base strategy we used. The *20x7* and *12* strategies was our entrants to the 2008 and 2007 AAAI Computer Poker Competitions respectively. The *14* strategy is a strategy we considered entering in the 2008 competition, but it was ultimately superseded by the smaller *20x7*.

Opponent	Relative Size	Estimated Expected Value (mb/h)	95% Confidence Interval
20x32	2.53	2.1	±1.1
20x8 (Base)	1	14.5	±1.1
20x7	0.43	17.6	±4.4
14	0.82	13.3	±1.8
12	0.45	18.0	±3.0

Table 5.3: Grafted Strategy against equilibrium-like opponents in Texas Hold'em

From the results, we can see that the grafted strategy beats all of the players with statistical significance, even the largest single strategy. We note that the amount at which it beats the smaller strategies is much more impressive than the amount it beats the largest strategy. Previously, the Computer Poker Research Group has noticed that doubling the size of an abstract game gave approximately a 4 to 6 mb/h improvement over the previous generation. By this estimate, the grafted strategy should beat its base by 15 to 23 mb/h. The amount it beats its base strategy by is in this range, but it does not appear to beat the previous generations by increasing amounts. Of note, it actually beats *14* by less than it beats *20x7* even though *20x7* beats *14*.

From the results, we can gather that the base strategy is still quite an important factor, but one that can be overcome. Since the grafts are forced to mesh well with the base strategy, a weak base strategy may not allow the grafts to take full advantage of the extra information they have. For example, imagine you were to play Texas Hold'em with a younger, less informed sibling, under the agreement that you would play the first two betting rounds and your sibling the second two. You might deviate your strategy in a non-optimal fashion in order to compensate for the weaknesses of your sibling to achieve the best possible outcome. Due to this reliance on the base strategy, it is unlikely that strategy grafting can provide a overwhelming improvement in the quality of the resulting strategy, which is reiterated in the results. Despite this, the grafted strategy appears to beat strategies created with current methods due to the sheer size of the strategies we can now compute.

From this, we see that strategy grafting is a competitive idea that allows one to augment their existing strategies to create even stronger players. Any improvement to the quality of a base strategy should in turn improve the quality of the grafted strategy. This means that strategy grafting can be used transparently on top of more sophisticated frameworks for finding good strategies.

Chapter 6

Conclusion

In Chapter 3, we observed that abstraction pathologies are prevalent in the the variants of Leduc Hold'em we examined. That is, after we formalized Assumption 1, we found that by refining and then solving an abstract game, and thus giving the players more information on which to base their decisions, we could arrive at a worse approximation of an equilibrium than if we had not refined said game. Some of the individual counterexamples were quite disturbing. In particular, the best strategy in the biggest abstract game where both players were abstracted, *J.Q.K-J.Q.K*, was one of the more exploitable strategies. That is, many of the smaller abstract games provided us with much better strategies in terms of exploitability. Another troubling fact was that the best strategies in terms of exploitability in these abstract games were typically quite terrible themselves or far from the exploitability of the arbitrary strategy found using CPLEX. Furthermore, these results were not the fault of the strategy spaces themselves. For example, even the worst strategy space, the Leduc abstraction that could not distinguish its preflop cards, had a strategy that ranked fourth best out of all the strategies we tested. Since the abstractions we used in Leduc Hold'em were similar to those used in the large poker games, we can say with confidence that it is likely these pathologies exist in the larger games.

This evidence will not stop us from abstracting games in the future. We have little choice but to employ its use in the large games of interest if we wish to make use of our extensive game solving algorithms. In poker in particular, these techniques seem to perform quite well, even in light of these pathologies. That is, the strategies found by solving abstract games are far stronger, against both human and computer opponents, than any of the previous techniques. Until a technique is developed to skillfully identify and exploit the weaknesses in these abstract equilibrium strategies, we are likely safe to continue with our current methods.

In Chapter 4, we explained how the superior performance of the larger strategies by further analyzing our Leduc Hold'em strategies. We first observed that our Leduc Hold'em strategies followed a similar pattern to the AAAI Computer Poker Competition when evaluated using the same tournament formats. That is, even though many of the larger abstract games produced more exploitable strategies, these strategies still outperformed the strategies from the smaller abstract games. None

of these strategies attempt to exploit their opponents; they merely benefit from their opponent's mistakes. This is a trait shared by most of the strong programs in the AAAI competitions. With this observation, we introduced the domination value to better measure performance in this setting. The domination value allowed us to quantitatively measure the amount a strategy might lose due to mistakes in its play. By using the domination value, as opposed to exploitability, to analyze the Leduc Hold'em strategies, we found a strong correlation between the outcome of the Leduc tournaments and the domination value of the strategies. Even though pathologies exist when using the domination value as the evaluation criteria, the strategies from the larger abstract games often to had smaller domination values than the strategies from the smaller abstract games. Furthermore, the domination value for an arbitrary abstract equilibrium strategy appeared to be close to that of best possible domination value in that abstract game, which is of high practical value. This evidence allowed us to conclude that using larger abstract games tends to lead to strategies that make fewer mistakes, and hence do better against non-exploitive opponents.

It was this observation that allows us to be confident in strategy grafting, a technique we introduced in Chapter 5 that can have a potentially negative effect on exploitability. Previous attempts to decompose extensive games with imperfect information into sub-games have failed to improve upon solving a single abstract game. This is most likely due to the fact that all parts of a strong strategy work together. Strategy grafting forces each sub-game strategy to conform to a common base strategy. We observed that this conformance led to a strong strategy when the sub-game strategies were combined as evident by the typical decrease in domination value. That is, the grafted strategies performed well in both the large and small poker games in tournament settings and improved on the state-of-the-art from 2008.

6.1 Future Work

There are a few directions in which we can move forward with these new results. First, it is possible to compute the best response value for a strategy in the full game of Texas Hold'em as the program can be easily distributed. A rough estimate is that it would take approximately six months on a powerful cluster of computers to compute a single best response value. We cannot compute the exploitability of a single strategy without knowing the value of the game, which we cannot compute for Texas Hold'em, but since the game is zero-sum we can compute the exploitability for a strategy profile. This has multiple implications. First, it could verify our suspicions that these equilibrium-like strategies are indeed quite exploitable. Second, it can tell us how far from equilibrium our current strategies are. Third, by computing this for two different abstractions it could potentially verify that the non-monotonicities are indeed present in the large game. For these reasons, this computation is clearly of interest, but it is unlikely it answers any pressing questions. Unfortunately, unlike the exploitability of strategy profile, the domination value of a strategy or a strategy profile cannot be feasibly computed in the full game.

A second, more promising direction, would be to further explore the effect of dominated strategies in these games. The equilibrium-like strategies are attempting to hide information in an effort to maintain low exploitability. We now know that they are not performing this task incredibly well (at least against a powerful adversary). We are now left to question what effect this information hiding has on their performance. This is something that neither exploitability nor domination value can individually measure. That is, exploitability measures both the ability to hide information as well as avoid making errors, whereas domination value measures only the ability to avoid errors. The effect that information hiding has on a strategy's performance does potentially depend on the opponent. For example, an expert human opponent that can quickly adapt would likely beat a program that plays flawlessly, but does not hide information. An equilibrium strategy, though, would tie a strategy that plays flawlessly, but fails to hide information.

If we are willing to stray from our goal of approximating an equilibrium, as it appears we should, we can investigate techniques that we know can potentially produce lower quality strategies in the worst-case, like strategy grafting. One such technique along these lines that could be reinvestigated would be to use a regret-minimizing learner during play. We know that such a learner will eventually converge on a best response strategy against any opponent, but a major problem is that matches are far too short to ever get close to this best response. That is, we are better off computing an equilibrium ahead of time than trying to learn an exploitive strategy during play in our tournament settings. One potential way to alleviate this concern is to bootstrap the learner with a good strategy ahead of time (*e.g.*, an equilibrium-like strategy). If we restrict the learner by only allowing it to take non-dominated actions, then we can potentially learn an exploitive strategy much more quickly while avoiding costly mistakes. Moreover, this learner should have similar domination value to the original base strategy as the two share a common action space. One drawback is that though the learner can only make the same mistakes as the base strategy, it could potentially make these mistakes more frequently.

An alternative approach along the same lines would be to deterministically take the action that maximizes one's expected value, given some beliefs about the opponent, without considering dominated actions. As opponent modeling is a hard task, it is likely that the beliefs about an opponent will be wrong in some fashion, but as long as we are not making too many mistakes, one would presume that this approach could do well, at least against equilibrium-like opponents.

Bibliography

[1] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a Rigged Casino: The Adversarial Multi-arm Bandit Problem. In *36th Annual Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[2] 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 *International Joint Conference on Artificial Intelligence*, pages 661–668, 2003.

[3] Darse Billings. RoShamBo Programming Competition, 2001. `http://www.cs.ualberta.ca/~darse/rsbpc.html`.

[4] D. Blackwell. An analog of the Minimax Theorem for Vector Payoffs. *Pacific Journal of Mathematics*, 6:1–8, 1956.

[5] V. Conitzer and T. Sandholm. Complexity of (Iterated) Dominance. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pages 88–97, 2005.

[6] Andrew Gilpin, Samid Hoda, Javier Peña, and Tuomas Sandholm. Gradient-based Algorithms for Finding Nash Equilibria in Extensive Form Games. In *Proceedings of the Eighteenth International Conference on Game Theory*, 2007.

[7] Andrew Gilpin and Tuomas Sandholm. A Competitive Texas Hold'em Poker poker via Automated Abstraction and Real-time Equilibrium Computation. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[8] Andrew Gilpin and Tuomas Sandholm. 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 Press, 2007.

[9] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sorensen. A heads-up no-limit Texas Hold'em Poker Player: Discretized Betting models and Automatically Generated Equilibrium-finding Programs. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2008.

[10] S. Hart and A. Mas-Colell. A General Class of Adaptive Strategies. *Journal of Economic Theory*, 1:26–54, 2001.

[11] Inc. Ilog. Solver CPLEX, 2009. `http://www.ilog.com/products/cplex/`.

[12] Michael Johanson. Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. Master's thesis, University of Alberta, 2007.

[13] Daphne Koller and Avi Pfeffer. Representations and Solutions for Game-Theoretic Problems. *Artificial Intelligence*, 94:167–215, 1997.

[14] Y. Nesterov. Excessive Gap Technique for Nonsmooth Convex Optimization. *SIAM Journal of Optimization*, 16(1):235–249, 2005.

[15] John Von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[16] M. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[17] David Schnizlein, Michael Bowling, and Duane Szafron. Probabilistic State Translation in Extensive Games with Large Action Sets. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 276–284, 2009.

[18] 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 21st Annual Conference on Uncertainty in Artificial Intelligence*, pages 550–558. AUAI Press, 2005.

[19] Bernhard Von Stengel. Efficient Computation of Behavior Strategies. *Games and Economic Behavior*, 14:220–246, 1996.

[20] Kevin Waugh, Nolan Bard, and Michael Bowling. Strategy Grafting in Extensive Games. In *Advances in Neural Information Processing Systems 22*, 2010. To appear.

[21] Kevin Waugh, Dave Schnizlein, Michael Bowling, and Duane Szafron. Abstaction Pathologies in Extensive Games. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 781–788, 2009.

[22] 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*, pages 788–793, 2007.

[23] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems 20*, pages 1729–1736, 2008.

[24] Martin Zinkevich and Michael Littman. The AAAI Computer Poker Competition. *Journal of the International Computer Games Association*, 29, 2006. News item.

Appendix A

Crosstables

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)	(25)	Average
(1)		14	34	-26	19	34	27	16	16	42	23	174	154	36	148	77	164	124	109	249	164	155	158	166	301	99.12
(2)	-14		52	-19	66	66	64	54	72	76	71	132	126	73	90	152	72	82	79	189	114	112	119	120	177	88.56
(3)	-34	-52		-25	60	63	47	41	104	106	88	110	92	85	67	176	46	49	60	146	106	111	117	115	177	77.32
(4)	26	19	25		12	43	21	32	29	39	49	79	76	48	82	64	53	73	84	89	146	140	141	150	258	73.98
(5)	-19	-66	-60	-12		34	-7	41	46	41	46	118	114	68	78	145	52	53	92	143	135	142	151	148	239	71.83
(6)	-34	-66	-63	-43	-34		-18	8	-20	12	27	140	119	36	111	58	88	76	91	186	149	143	141	151	316	65.62
(7)	-27	-64	-47	-21	7	18		16	17	13	24	84	105	39	43	92	46	45	67	119	106	102	113	115	191	50.19
(8)	-16	-54	-41	-32	-41	-8	-16		-15	-4	11	77	115	9	58	66	23	41	67	126	95	94	108	106	251	42.46
(9)	-16	-72	-104	-29	-46	20	-17	15		10	29	101	88	34	62	80	15	40	58	113	90	93	96	100	191	39.62
(10)	-42	-76	-106	-39	-41	-12	-13	4	-10		25	54	39	21	20	58	30	38	45	69	100	97	102	106	177	26.89
(11)	-23	-71	-88	-49	-46	-27	-24	-11	-29	-25		63	55	5	56	15	37	42	48	89	88	84	80	95	209	23.93
(12)	-174	-132	-110	-79	-118	-140	-84	-77	-101	-54	-63		49	-14	12	-21	29	46	76	98	182	182	182	184	313	7.68
(13)	-154	-126	-92	-76	-114	-119	-105	-115	-88	-39	-55	-49		-46	-4	-19	37	38	68	47	200	200	200	200	348	5.57
(14)	-36	-73	-85	-48	-68	-36	-39	-9	-34	-21	-5	14	46		14	-6	-7	19	41	59	58	56	63	69	124	4.11
(15)	-148	-90	-67	-82	-78	-111	-43	-58	-62	-20	-56	-12	4	-14		9	42	25	46	75	112	100	111	118	244	1.87
(16)	-77	-152	-176	-64	-145	-58	-92	-66	-80	-58	-15	21	19	6	-9		-10	38	61	114	141	144	144	146	201	1.41
(17)	-164	-72	-46	-53	-52	-88	-46	-23	-15	-30	-37	-29	-37	7	-42	10		14	48	5	88	107	117	104	69	-7.00
(18)	-124	-82	-49	-73	-53	-76	-45	-41	-40	-38	-42	-46	-38	-19	-25	-38	-14		18	9	51	45	49	59	82	-22.18
(19)	-109	-79	-60	-84	-92	-91	-67	-67	-58	-45	-48	-76	-68	-41	-46	-61	-48	-18		-49	26	31	33	43	64	-42.14
(20)	-249	-189	-146	-89	-143	-186	-119	-126	-113	-69	-89	-98	-47	-59	-75	-114	-5	-9	49		149	149	149	149	248	-42.98
(21)	-164	-114	-106	-146	-135	-149	-106	-95	-90	-100	-88	-182	-200	-58	-112	-141	-88	-51	-26	-149		10	7	20	-29	-95.49
(22)	-155	-112	-111	-140	-142	-143	-102	-94	-93	-97	-84	-182	-200	-56	-100	-144	-107	-45	-31	-149	-10		-3	12	-32	-96.76
(23)	-158	-119	-117	-141	-151	-141	-113	-108	-96	-102	-80	-182	-200	-63	-111	-144	-117	-49	-33	-149	-7	3		13	-35	-99.98
(24)	-166	-120	-115	-150	-148	-151	-115	-106	-100	-106	-95	-184	-200	-69	-118	-146	-104	-59	-43	-149	-20	-12	-13		-42	-105.39
(25)	-301	-177	-177	-258	-239	-316	-191	-251	-191	-177	-209	-313	-348	-124	-244	-201	-69	-82	-64	-248	29	32	35	42		-168.25

Figure A.1: Crosstable for Leduc Hold'em Bankroll Tournament

Rank	Strategy Name
FULL-J.QK	1
FULL-J.Q.K	2
FULL-JQ.K	3
FULL-FULL	4
J.Q.K-JQ.K	5
J.Q.K-J.QK	6
J.Q.K-J.Q.K	7
J.Q.K-FULL	8
JQ.K-JQ.K	9
JQ.K-J.Q.K	10
JQ.K-J.QK	11
J.Q.K-JQK	12
FULL-JQK	13
JQ.K-FULL	14
J.QK-J.QK	15
JQ.K-JQK	16
J.QK-JQ.K	17
J.QK-J.Q.K	18
J.QK-FULL	19
J.QK-JQK	20
JQK-JQ.K	21
JQK-FULL	22
JQK-JQK	23
JQK-J.QK	24

Table A.1: Legend for Table A.1

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	(21)	(22)	(23)	(24)	(25)	(26)	(27)	(28)	(29)	(30)	(31)	(32)	Average
(1)		14	34	-26	5	7	19	20	34	6	45	63	27	16	16	58	42	23	36	94	148	154	174	164	77	124	109	249	155	164	166	301	81.24
(2)	-14		52	-19	30	33	66	31	66	57	38	-21	64	54	72	36	76	71	73	29	90	126	132	72	152	82	79	189	112	114	120	177	72.23
(3)	-34	-52		-25	22	21	60	18	63	94	46	-11	47	41	104	20	106	88	85	49	67	92	110	46	176	49	60	146	111	106	115	177	64.43
(4)	26	19	25		10	13	12	45	43	24	23	44	21	32	29	52	39	49	48	88	82	76	79	53	64	73	84	89	140	146	150	258	62.39
(5)	-5	-30	-22	-10		2	27	12	48	28	18	77	27	29	41	22	38	42	48	91	57	123	109	78	117	61	81	136	112	119	127	171	57.19
(6)	-7	-33	-21	-13	-2		22	17	48	29	19	71	24	24	43	25	42	39	47	94	51	117	100	81	114	56	77	127	110	118	127	172	55.40
(7)	-19	-66	-60	-12	-27	-22		-20	34	29	5	73	-7	41	46	17	41	46	68	64	78	114	118	52	145	53	92	143	142	135	148	239	54.55
(8)	-20	-31	-18	-45	-12	-17	20		8	-67	11	52	6	3	-13	31	16	0	18	102	113	120	122	106	39	69	78	222	103	114	126	283	49.67
(9)	-34	-66	-63	-43	-48	-48	-34	-8		-80	-44	75	-18	8	-20	44	12	27	36	118	111	119	140	88	58	76	91	186	143	149	151	316	46.53
(10)	-6	-57	-94	-24	-28	-29	-29	67	80		-47	49	-1	27	32	80	36	44	50	87	90	111	126	22	110	53	67	121	84	86	101	200	45.46
(11)	-45	-38	-46	-23	-18	-19	-5	-11	44	47		14	9	14	71	4	87	50	43	14	66	71	116	59	134	20	42	91	93	104	111	214	42.38
(12)	-63	21	11	-44	-77	-71	-73	-52	-75	-49	-14		-63	-63	-27	-12	35	23	-3	187	35	65	78	80	155	45	74	127	178	187	180	491	41.48
(13)	-27	-64	-47	-21	-27	-24	7	-6	18	1	-9	63		16	17	8	13	24	39	81	43	105	84	46	92	45	67	119	102	106	115	191	38.01
(14)	-16	-54	-41	-32	-29	-24	-41	-3	-8	-27	-14	63	-16		-15	13	-4	11	9	136	58	115	77	23	66	41	67	126	94	95	106	251	33.12
(15)	-16	-72	-104	-29	-41	-43	-46	13	20	-32	-71	27	-17	15		34	10	29	34	50	62	88	101	15	80	40	58	113	93	90	100	191	25.56
(16)	-58	-36	-20	-52	-22	-25	-17	-31	-44	-80	-4	12	-8	-13	-34		8	-12	9	75	86	28	58	83	29	55	66	113	104	116	125	271	25.32
(17)	-42	-76	-106	-39	-38	-42	-41	-16	-12	-36	-87	-35	-13	4	-10	-8		25	21	30	20	39	54	30	58	38	45	69	97	100	106	177	10.05
(18)	-23	-71	-88	-49	-42	-39	-46	-0	-27	-44	-50	-23	-24	-11	-29	12	-25		5	-9	56	55	63	37	15	42	48	89	84	88	95	209	9.63
(19)	-36	-73	-85	-48	-48	-47	-68	-18	-36	-50	-43	3	-39	-9	-34	-9	-21	-5		52	14	46	14	-7	-6	19	41	59	56	58	69	124	-4.01
(20)	-94	-29	-49	-88	-91	-94	-64	-102	-118	-87	-14	-187	-81	-136	-50	-75	-30	9	-52		-41	-15	21	31	115	6	30	70	132	136	132	332	-15.68
(21)	-148	-90	-67	-82	-57	-51	-78	-113	-111	-90	-66	-35	-43	-58	-62	-86	-20	-56	-14	41		4	-12	42	9	25	46	75	100	112	118	244	-16.83
(22)	-154	-126	-92	-76	-123	-117	-114	-120	-119	-111	-71	-65	-105	-115	-88	-28	-39	-55	-46	15	-4		-49	37	-19	38	68	47	200	200	200	348	-22.08
(23)	-174	-132	-110	-79	-109	-100	-118	-122	-140	-126	-116	-78	-84	-77	-101	-58	-54	-63	-14	-21	12	49		29	-21	46	76	98	182	182	184	313	-23.45
(24)	-164	-72	-46	-53	-78	-81	-52	-106	-88	-22	-59	-80	-46	-23	-15	-83	-30	-37	7	-31	-42	-37	-29		10	14	48	5	107	88	104	69	-26.63
(25)	-77	-152	-176	-64	-117	-114	-145	-39	-58	-110	-134	-155	-92	-66	-80	-29	-58	-15	6	-115	-9	19	21	-10		38	61	114	144	141	146	201	-29.75
(26)	-124	-82	-49	-73	-61	-56	-53	-69	-76	-53	-20	-45	-45	-41	-40	-55	-38	-42	-19	-6	-25	-38	-46	-14	-38		18	9	45	51	59	82	-30.54
(27)	-109	-79	-60	-84	-81	-77	-92	-78	-91	-67	-42	-74	-67	-67	-58	-66	-45	-48	-41	-30	-46	-68	-76	-48	-61	-18		-49	31	26	43	64	-50.31
(28)	-249	-189	-146	-89	-136	-127	-143	-222	-186	-121	-91	-127	-119	-126	-113	-113	-69	-89	-59	-70	-75	-47	-98	-5	-114	-9	49		149	149	149	248	-70.56
(29)	-155	-112	-111	-140	-112	-110	-142	-103	-143	-84	-93	-178	-102	-94	-93	-104	-97	-84	-56	-132	-100	-200	-182	-107	-144	-45	-31	-149		-10	12	-32	-104.42
(30)	-164	-114	-106	-146	-119	-118	-135	-114	-149	-86	-104	-187	-106	-95	-90	-116	-100	-88	-58	-136	-112	-200	-182	-88	-141	-51	-26	-149	10		20	-29	-105.76
(31)	-166	-120	-115	-150	-127	-127	-148	-126	-151	-101	-111	-180	-115	-106	-100	-125	-106	-95	-69	-132	-118	-200	-184	-104	-146	-59	-43	-149	-12	-20		-42	-114.36
(32)	-301	-177	-177	-258	-171	-172	-239	-283	-316	-200	-214	-491	-191	-251	-191	-271	-177	-209	-124	-332	-244	-348	-313	-69	-201	-82	-64	-248	32	29	42		-200.25

Figure A.2: Crosstable of Leduc Hold'em Bankroll Tournament with Grafted Strategies

Rank	Name
FULL-J.QK	1
FULL-J.Q.K	2
FULL-JQ.K	3
FULL-FULL	4
J.Q.K-J.Q.K preflop grafts	5
J.Q.K-JQ.K	6
J.Q.K-J.Q.K flop grafts	7
J.QK-J.QK preflop grafts	8
J.Q.K-J.QK	9
JQ.K-JQ.K flop grafts	10
JQK-JQK preflop grafts	11
JQ.K-JQ.K preflop grafts	12
J.Q.K-J.Q.K	13
J.Q.K-FULL	14
J.QK-J.QK flop grafts	15
JQ.K-JQ.K	16
JQ.K-J.Q.K	17
JQ.K-J.QK	18
JQ.K-FULL	19
JQK-JQK flop grafts	20
J.QK-J.QK	21
FULL-JQK	22
J.Q.K-JQK	23
J.QK-JQ.K	24
JQ.K-JQK	25
J.QK-J.Q.K	26
J.QK-FULL	27
J.QK-JQK	28
JQK-JQ.K	29
JQK-FULL	30
JQK-JQK	31
JQK-J.QK	32

Table A.2: Legend for Table A.2