Using Response Functions for Strategy Training and Evaluation

by

Trevor Davis

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

Extensive-form games are a powerful framework for modeling sequential multi-agent interactions. In extensive-form games with imperfect information, Nash equilibria are generally used as a solution concept, but computing a Nash equilibrium can be intractable in large games. Instead, a variety of techniques are used to find strategies that approximate Nash equilibria. Traditionally, an approximate Nash equilibrium strategy is evaluated by measuring the strategy's worst-case performance, or exploitability. However, because exploitability fails to capture how likely the worst-case is to be realized, it provides only a limited picture of strategy strength, and there is extensive empirical evidence showing that exploitability can correlate poorly with one-on-one performance against a variety of opponents. In this thesis, we introduce a class of adaptive opponents called pretty-good responses that exploit a strategy but only have limited exploitative power. By playing a strategy against a variety of counter-strategies created with pretty-good responses, we get a more complete picture of strategy strength than that offered by exploitability alone. In addition, we show how standard no-regret algorithms can me modified to learn strategies that are strong against adaptive opponents. We prove that this technique can produce optimal strategies for playing against pretty-good responses. We empirically demonstrate the effectiveness of the technique by finding static strategies that are strong against Monte Carlo opponents who learn by sampling our strategy, including the UCT Monte Carlo tree search algorithm.

Preface

Portions of this thesis were previously published in “Using Response Functions to Measure Strategy Strength” at AAAI 2014 [9].

Acknowledgements

First, I want to thank my supervisor, **Dr. Michael Bowling**. His guidance and support were instrumental to me developing as a researcher and writer, and at the same time he gave me the freedom to explore topics which I found interesting. I am perhaps most grateful for his easygoing personality, which helped me stay calm even at the most stressful of times.

Second, I want to thank my peers in the Computer Poker Research Group. In particular, I thank **Neil Burch** for guiding me along my first complete research project, from formation of a research problem to design of an algorithmic solution and finally to writing a paper that was ultimately accepted at AAAI. That research project led to this thesis, which wouldn't exist in its current form without Neil's assistance. I also want to give special thanks to **Michael Johanson** and **Richard Gibson** for giving me invaluable advice and assistance with the CPRG code base when I was first starting my research. All of the CPRG - **Nolan Bard**, **Josh Davidson**, **Johnny Hawkin**, **Parisa Mazrooi**, **Dustin Morrill**, as well as **Dr. Christopher Archibald**, **Dr. Rob Holte**, and **Dr. Duane Szafron** - has provided a wonderful environment for doing research and for getting feedback on my ideas.

Third, I want to thank the **University of Alberta** and **Alberta Innovates - Technology Futures** for awarding me with generous scholarships that supported by graduate studies, allowing me to focus on my research. In addition, I thank **WestGrid** and **Compute Canada** for the computation resources that they provided me and the rest of the CPRG on their supercomputer clusters. Without their support, many of the experiments in this thesis wouldn't have been possible.

Finally, I want to thank my parents **Bruce** and **Mary**. Without their

support and encouragement, I never would have been able to pursue graduate studies in the first place.

Contents

List of Figures

Chapter 1

Introduction

Extensive-form games are a natural representation for sequential decision-making tasks. By incorporating multiple agents, stochastic outcomes, and hidden information, they provide a powerful framework for modeling real-world situations and also generalize other popular models of decision making such as simultaneous games and (finite-horizon) partially observable Markov decision processes. This generality makes extensive-form games valuable as a domain for developing and testing artificial intelligence techniques.

In extensive-form games with imperfect information, Nash equilibrium strategies are generally used as a solution concept. Efficient algorithms exist for finding Nash equilibria in two-player zero-sum games, but even with these tools finding Nash equilibria is intractable in the large games that often result from human interaction. For example, there exist commonly played poker variants in which directly applying the most efficient algorithms for finding Nash equilibria would require more bytes of RAM than there are atoms in the observable universe [22].

In order to create agents in these large games, a variety of techniques have been employed, often resulting in a static strategy that an agent can use to play the game with simple table lookups. The most successful method turns the large game into a smaller, abstract game by artificially preventing the players from observing some information, and then uses state-of-the-art algorithms to find a Nash equilibrium in the abstract game. One way to evaluate these techniques is to measure how close the resulting strategy is to a

Nash equilibrium for the full game, but this raises the question of how such a similarity should be measured, especially when no Nash equilibrium is known.

In two-player zero-sum games, Nash equilibrium strategies are guaranteed to minimize worst-case losses. In other words, if a player must face an adversarial opponent who both always knows her strategy and can perfectly respond to it to maximize his own expected utility, she cannot do better than to play a Nash equilibrium strategy. Because of this relation, the worst-case performance, or exploitability, of a strategy is often used to measure its distance from a Nash equilibrium and thus evaluate its strength [33, 24, 14].

Unfortunately, as Ganzfried et al. previously identified, there are conceptual issues with exploitability as an evaluation technique as well as computational limitations which prevent its use in very large games [14]. In addition, empirical results have shown limited correlation between exploitability and actual one-on-one performance [32, 24, 4]. This leads us to propose two related questions which motivate the work presented in this thesis:

1. How do we evaluate strategies in large extensive-form games?
2. Given an evaluation metric, how do we efficiently find a strategy that performs well when evaluated by the metric?

1.1 Contributions

This thesis makes the following contributions:

- A class of strategy evaluation metrics which make use of “pretty-good responses” to generate counter-strategies. These metrics generalize existing evaluation techniques such as exploitability and expected utility against a static opponent.
- A family of computational techniques called CFR-f which find strategies which are strong when evaluated by a particular response function f. When f is a pretty-good response, CFR-f converges to an optimal strategy as evaluated by f.

- Experimental results which demonstrate the effectiveness of evaluating strategies with our generalized metrics as well as the effectiveness of CFR-f at finding strategies which are strong against adaptive opponents.

1.2 Thesis Structure

In Chapter 2 of this thesis, we formalize the notion of extensive-form games, introduce poker games as testbeds for extensive-form game techniques, and present the necessary background in game theory and game-solving for us to motivate and present our original contributions.

In Chapter 3, we examine methods for evaluating strategies in extensive-form games. By looking at accumulated results from previous work, we argue that exploitability fails to offer a complete picture of strategy strength. We then propose a new class of evaluation metrics, which use techniques called pretty-good response functions to find a counter-strategy to the strategy being evaluated and measure the performance of the strategy against the counter-strategy. Instead of the absolute worst-case opponent used in calculating exploitability, pretty-good responses generate "bad-case" opponents for different degrees of "bad." By comparing the results of different pretty-good response evaluations, we can get a picture of not only a strategy's worst-case performance, but also how often the worst case is realized; in other words, not just how exploitable a strategy is, but how difficult it is for an opponent to exploit it. We conclude by showing empirically that pretty-good responses can measure dimensions of strategy strength not captured by exploitability.

In Chapter 4, we turn to question #2 and look at using our new evaluation metrics to learn strong strategies. Any regret minimization algorithm can be modified such that the opponent formulates his strategy as a response to the player's (regret-minimizing) strategy. When the adaptive agent is a pretty-good response, we prove that the regret minimization algorithm will converge to a strategy that is optimal when evaluated by the pretty-good response. We further prove a bound on the optimality of the resulting strategy for the cases where we can bound the adaptive player's deviation from a pretty-good response, including for stochastic responses. Finally, we show how an existing framework for creating strategies that make a tradeoff between exploitation and exploitability can be generalized using pretty-good responses.

In Chapter 5, we empirically test our algorithm for finding static strategies which are optimal against adaptive opponents. In particular, we train against opponents that use Monte Carlo sampling techniques to learn how to play well against our strategy. In a small poker game, we learn to play against UCT, a Monte Carlo tree search algorithm [28] which is used in strong agents in complex games such as Go [15]. We show that we are able to generate strategies which perform better against UCT opponents than a Nash equilibrium does against the same opponents. We also test how our technique scales to large poker games by training against Frequentist Best-Responses, a Monte Carlo technique for quickly approximating a best-response which has been previously applied to poker [23, 21].

Chapter 2

Background

In this chapter we will present the notation and concepts that will be used in this thesis, and we will examine some previous work that our contributions build on. In section 2.1 we formalize the notion of extensive-form games, and in section 2.2 we describe the poker games that we will use in this thesis to test our techniques. Sections 2.3, 2.4, 2.5, and 2.6 describe solution concepts and algorithms used in previous work.

2.1 Extensive-Form Games

We can model any finite sequential, multi-agent interaction by using a game tree. Each node represents a state of the game, with interaction beginning at the root. At each internal node of the tree, one of the agents must choose an action, represented by an edge, which results in a new game state corresponding to the child node. At the tree's leaves, the game ends with each agent assigned a utility for this outcome. Stochastic events in the game are represented by chance nodes, where each edge is taken with a predefined probability.

In many interactions of note, the agents can't perfectly observe the state of the game. For example, in poker games a player cannot observe the cards held by her opponent, which are the outcome of a chance node. Games of this type are said to exhibit **imperfect information**. We model them by partitioning the game states of an extensive-form game into **information sets**, where the acting player can only distinguish two game states if they are in different information sets. We can model this formally as an **extensive-form game**.

Definition 1. *A finite* **extensive form game** Γ *consists of:*

- *A finite set N of* **players***.*
- *A finite set H of* **histories***. H contains sequences of actions. For $h, h' \in H$, we write $h \sqsubseteq h'$ if h is a prefix of h'. For all sequences $h \in H$, any $h_0 \sqsubseteq h$ is also in H. Thus H contains the empty sequence. Define $Z \subseteq H$ to be the set of* **terminal histories***. Thus $z \in Z$ if and only if there is no $h \in H$ such that $z \sqsubset h$ ($z \sqsubseteq h$ and $z \neq h$). For $h \in H$, define $A(h) = \{a : (h, a) \in H\}$ to be the set of actions available at h, where (h, a) is the sequence that appends a to h. $A(h)$ is the* **action set***.*
- *A* **player function** $P\colon H \backslash Z \to N \cup \{c\}$*. $P(h)$ is the player who chooses the action at h. If $P(h) = c$ then chance chooses the action. We define $H_i = \{h \in H : P(h) = i\}$ for all $i \in N \cup \{c\}$.*
- *A* **chance function** σ_c *which assigns a probability distribution over actions at every $h \in H_c$. We write $\sigma_c(\cdot|h)$ for the distribution and $\sigma_c(a|h)$ for the probability that $a \in A(h)$ occurs at history h. Since $\sigma_c(\cdot|h)$ is a probability distribution, $\sum_{a \in A(h)} \sigma_c(a|h) = 1$.*
- *For each player $i \in N$, a* **utility function** $u_i\colon Z \to \mathbb{R}$*. When the game reaches a terminal history $z \in Z$, each player i receives payoff $u_i(z)$ and the game is over. If $N = \{1, 2\}$ and $u_1(z) = -u_2(z)$ for all $z \in Z$ then we call the game* **zero-sum***.*
- *For each player $i \in N$, an* **information partition** $\mathcal{I}_i$ *of H_i. If $I_i \in \mathcal{I}_i$ we call I_i an* **information set** *for player i. The partition must satisfy the property that if $h, h' \in I_i$, then $A(h) = A(h')$. We define $A(I_i) = A(h)$ and $P(I_i) = i$.*

Any information partition results in a legal extensive-form game, but many partitions result in unnatural models where a player can know something about the current game state, but then forget that information in a later game state. To avoid these complications, we assume an additional property known as **perfect recall**, which requires that no player forgets information. For every history h where there is some $g \sqsubset h$ such that $P(g) = P(h)$, define $D(h)$ to be the last predecessor of h where $P(h)$ acted. Formally, let $D(h)$ be the unique history such that $D(h) \sqsubset h$, $P(D(h)) = P(h)$ and there is no g such that $D(h) \sqsubset g \sqsubset h$ and $P(g) = P(h)$.

Definition 2. *An extensive form game exhibits* **perfect recall** *if for any two histories h, h' in the same information set I, either $D(h)$ and $D(h')$ are both undefined, or they are both defined and*

- *$D(h), D(h') \in I$ for some $I \in \mathcal{I}_i$ where $i = P(h)$.*
- *For any $a \in A(D(h))$, $(D(h), a) \sqsubseteq h$ if and only if $(D(h'), a) \sqsubseteq h'$.*

If an extensive form game does not exhibit perfect recall, it is said to exhibit **imperfect recall**.

By inducting backward, we see that perfect recall implies that a player remembers each of his previous information sets, as well as the action she took at each information set. This thesis will only consider games which exhibit perfect recall.

2.1.1 Strategies

Informally, a strategy is a player's plan for how to play a game. A **pure strategy** specifies a single deterministic action at each of a player's decision points. For an extensive-form game, this takes the form of a function that maps each of a player's information sets to a legal action at that information set. A **mixed strategy** is a probability distribution over pure strategies. In extensive-form games with perfect recall, there is a one-to-one correspondence between mixed strategies and **behavioral strategies**, which specify a probability distribution over actions at every information set.

Definition 3. *We call σ_i a* **behavioral strategy for player** *i if $\sigma_i(\cdot|I_i)$ is a probability distribution over $A(I_i)$ for every $I_i \in \mathcal{I}_i$. We denote the probability that σ_i chooses $a \in A(I_i)$ at information set I_i by $\sigma_i(a|I)$. We label the set of all behavioral strategies for player i as Σ_i.*

Because any pure or mixed strategy for a perfect recall extensive-form game can be represented as a behavioral strategy, this thesis will deal strictly with behavioral strategies unless otherwise specified. When we refer to a **strategy**, we specifically mean a behavioral strategy.

A set of strategies for each player $i \in N$ is called a **strategy profile** and labelled σ. We label the set of all strategy profiles as Σ. The set of strategies in σ except for that of player i is labelled σ_{-i}. We define $u_i(\sigma)$ to be the expected utility of player i if all players play according to σ, and $u_i(\sigma_i, \sigma_{-i})$ to be the expected utility for player i if she plays according to σ_i and all other players play according to σ_{-i}.

Define $\pi^\sigma(h)$ to be the **reaching probability** for history h. This is equal to the probability of history h occuring if all players play according to $\sigma \in \Sigma$. This probability can be decomposed $\pi^\sigma(h) = \prod_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ where $\pi_i^\sigma(h)$ is the probability that player i takes the necessary actions to reach h, irrespective of the other players. Define $\pi_{-i}^\sigma(h)$ to be the contribution to $\pi^\sigma(h)$ from chance and all players except player i. Also define $\pi^\sigma(h, h')$ to be the probability of reaching h' from h.

Define $\overline{\sigma}^T$ to be the strategy profile that averages over the profiles $\sigma^1, ..., \sigma^T$. This can be defined at each information set:

Definition 4. [36] *The* **average strategy** *for player i is*

$$\overline{\sigma}_i^T(a|I) = \frac{\sum_{t=1}^T \pi_i^{\sigma^t}(I)\sigma^t(a|I)}{\sum_{t=1}^T \pi_i^{\sigma^t}(I)} \tag{2.1}$$

where $\pi_i^{\sigma^t}(I)$ is the compound probability that player i takes each of the actions necessary to reach I (under σ_i^t).

It is easy to show that in perfect recall games, for any set of opponent strategies, this average strategy has expected utility equal to the mixed strategy that randomly selects between each of $\sigma^1, ..., \sigma^T$ at the start of the game. This can be stated formally as

$$\frac{1}{T}\sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}) = u_i(\overline{\sigma}_i^T, \sigma_{-i}) \tag{2.2}$$

2.2 Poker Games

When we design techniques for extensive-form games, we need example games in which to test the techniques. For much of the research community, poker games fulfill this purpose, and this thesis will also make use of them. Poker games have several key advantages:

- Poker games incorporate chance events (dealing cards from a random deck) and imperfect information (hands of cards each visible to only one player). They thus make use of extensive-game features that don't exist in other common games like chess and go.

- Poker games allow for a range of payoffs. This means that poker agents must not only be concerned with whether they beat their opponent(s), but also with how much they win from them. This opens research avenues to create poker agents which are specifically designed to exploit certain opponents, which is a feature not found in other extensive-form games like liar's dice.

- Poker games are commonly played by humans. This means that there is a wealth of domain information for researchers to draw on, there is public interest in poker-related research, and researchers can test their agents by playing against human experts.

- Poker games are easily customizable depending on the focus of the technique being tested. The number and branching factor of chance events can be changed by modifying the size of the deck and the number of cards dealt, the number and branching factor of player nodes can be changed by modifying the number of betting rounds and the set of legal bet sizes, and the games can be easily extended to allow for any number of players.

In this thesis, we will concentrate on two poker games: **Texas Hold'em** and **Leduc Hold'em**. We will concern ourselves with the versions of these games that are **two-player** (also known as **heads-up**) and **(fixed)-limit**, which means that the number of bets per round is limited and all bets are of a predefined size.

2.2.1 Texas Hold'em

Two-player limit Texas Hold'em is the smallest poker game commonly played by humans. Because of this, it has been the focus of the majority of artificial intelligence research in poker so far, and has been used as a format in the Annual Computer Poker Competition since its inception in 2006 [1].

Texas Hold'em is played with a standard fifty-two card deck of playing cards, containing cards of thirteen different **ranks** in each of four **suits**. At the end of each discrete game (called a **hand** in poker terminology), each player will hold two private cards (also called a hand) and there be will five **community cards** face up and visible to each player. Each player forms the best possible five card poker hand from among the seven cards comprised of his private cards and the community cards. Poker hands define a total ordering (with ties) on the set containing all possible subsets of five cards. Whichever player has the hand which ranks highest by this ordering wins the **pot**, which is the sum total of all chips bet during the game. If there is a tie, the players evenly split the pot.

The actions during Texas Hold'em take place over a series of five **betting rounds**. Prior to the first betting round (called the **preflop**), the private cards are dealt (two per player). Prior to the second betting round (called the **flop**), the first three community cards are dealt. Prior to each of the third and fourth betting rounds (called the **turn** and **river** respectively), one additional community card is dealt.

At the start of each game, one player is designated as the **dealer**. In the two-player game, the dealer is also called the **small blind** and his opponent is called the **big blind**. At the start of the first betting round, the small blind places a forced bet equal to half of the predefined bet size, and the big blind places a forced bet equal to the total predefined bet size. During the first betting round, action starts with the small blind, while during the rest, it starts with the big blind. When faced with a bet, a player has three legal decisions: they can **fold**, which forfeits the hand, they can **call**, which requires them to place chips into the pot matching the bet, or they can **raise**, which requires them to place chips into the pot matching the current bet plus an extra bet that they make which the opponent will now be required to respond to. When a player is not facing a bet, they have two legal actions: they can **check**, in which they decline the opportunity to take an action, or they can **bet**, which requires them to place chips into the pot to make a bet that must be matched by the opponent. Because they are never possible at the same decision point, we often identify check with call and raise with bet.

On each betting round, a maximum of four bet/raise actions can be made between the players; after this limit has been reached, only fold and call are legal actions. On the first betting round, the big blind's forced bet counts as the first of these bets. A betting round ends after each player has had at least one opportunity to act and the last action was a call/check. All bets/raises on a particular betting round are forced to be the same size. On the preflop and flop betting rounds, this size is equal to the size of the big blind's forced bet. On the turn and river betting rounds, the size doubles to twice the big blind's forced bet.

2.2.2 Leduc Hold'em

Leduc Hold'em [31] is a small poker game designed to retain many of the strategic complexities of Texas Hold'em, while allowing researchers to quickly test techniques that either don't scale to Texas Hold'em or would require more computational resources than are available. In Leduc Hold'em, the deck contains six cards, comprising three ranks and two suits. Only one private card is dealt to each player, and only one community card is dealt. The best two card poker hand wins, with a pair beating a non-pair, and highest card deciding the winner otherwise.

At the start of a game of Leduc Hold'em, each player places a forced bet of one chip called the **ante** and is dealt his private card. There is then a betting round (called the preflop) which is identical to a Texas Hold'em betting round except that the maximum number of bets/raises is two. Then the community card is dealt and another betting round (called the flop) takes place, after which the game is over. The bet size is two chips for the preflop round and four chips for the flop round. Like in Texas Hold'em, one player is identified as the dealer. On each of the two betting rounds, the non-dealer player acts first.

2.3 Nash Equilibria

In an extensive-form game, a player's goal is to maximize her expected utility. If she knows the strategies that each of her opponents will play, she can do this by playing a **best-response** strategy, which is a strategy that achieves maximal expected performance against a particular set of opponent strategies.

Definition 5. *$\sigma_i \in \Sigma_i$ is a* **best-response** *to σ_{-i} if*

$$u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i}), \forall \sigma'_i \in \Sigma_i$$

In general, a player will not know her opponent's strategies, so she cannot directly maximize her expected utility. Instead, our solution concept is a **Nash equilibrium**, which assumes that each player is rational and will attempt to maximize her own expected utility, and thus requires that each player play a best-response to her opponents.

Definition 6. *A* **Nash equilibrium** *is a strategy profile $\sigma \in \Sigma$ where*

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

An **ε-Nash equilibrium** *is a an approximation of a Nash equilibrium where*

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

A strategy profile is a Nash equilibrium if and only if no player can gain utility by unilaterally changing his strategy, and is an ε-Nash equilibrium if and only if no player can gain more than ε utility by unilaterally changing his strategy.

2.3.1 Two-Player Zero-Sum Games

All games considered in this thesis will have two players and a zero-sum utility function. In such games, all Nash equilibria have the same expected utility, which is known as the **game value** and is given by the **minimax theorem**.

Theorem 1 (Minimax Theorem). *In any finite zero-sum game with two players*

$$v_1 = \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2) = \min_{\sigma_1 \in \Sigma_1} \max_{\sigma_2 \in \Sigma_2} u_1(\sigma_1, \sigma_2).$$

v_1 *is called the* **game value** *for player* 1*.*

Because the game is two-player and zero-sum, $v_2 = -v_1$ is the game value for player 2. If σ is a Nash equilibrium in a two-player zero-sum game, then $u_i(\sigma) = v_i$ for each player.

In a two-player zero-sum game, the **exploitability** of a strategy is how much expected utility a best-response opponent can achieve above the game value:

$$\begin{aligned} \text{exploit}(\sigma_i) &= \max_{\sigma^*_{-i} \in \Sigma_{-i}} u_{-i}(\sigma_i, \sigma^*_{-i}) - v_{-i} \\ &= v_i - \min_{\sigma^*_{-i} \in \Sigma_{-i}} u_i(\sigma_i, \sigma^*_{-i}). \end{aligned}$$

In large games, the game value may be intractable to compute, so we instead consider the average exploitability of a strategy profile:

$$\begin{aligned} \text{exploit}(\sigma) &= \frac{1}{2} \left(\text{exploit}(\sigma_1) + \text{exploit}(\sigma_2)\right) \\ &= \frac{1}{2} \left(\max_{\sigma^*_2 \in \Sigma_2} u_2(\sigma_1, \sigma^*_2) + \max_{\sigma^*_1 \in \Sigma_1} u_1(\sigma^*_1, \sigma_2) \right). \end{aligned}$$

Because each strategy in a Nash equilibrum profile is a best-response to the opponent, by definition $\text{exploit}(\sigma) = 0$ for any Nash equilibrium σ.

To adopt a convention, we will use female pronouns for player 1 and male pronouns for player 2 whenever we talk about a two-player game.

2.4 Regret and Finding Equilibria

Consider a two-player extensive-form game that is played over a series of time steps $t = 1, ..., T$. Let σ^t be the strategy profile played by the players at time t such that player i will receive expected utility $\sum_{t=1}^{T} u_i(\sigma^t)$ over the total time steps of the game. Because the other player's strategies σ_{-i}^t are not known, neither player knows which strategy is optimal to play against her opponent. We can evaluate a player's performance under these conditions using the concept of **regret**, which measures how much expected utility she lost by not playing the optimal strategy.

Definition 7. [36] *The* **average overall regret** *of player i at time T is*

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} \left(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t) \right) \tag{2.3}$$

This is the amount of utility that player i lost by playing according to σ^t for $t = 1, ..., T$ instead of playing the best static strategy σ_i^*. Minimizing regret is clearly connected to maximizing expected utility, so it's unsurprising that each player minimizing his regret can lead to a Nash equilbrium. The exact relation is given by a well-known theorem of unknown origin:

Theorem 2. *In a two player zero-sum game played for T iterations, with strategy profile σ^t played at time t, if $R_i^T \leq \varepsilon_i$ for players $i = 1, 2$, then $\overline{\sigma}^T$ is a $(\varepsilon_1 + \varepsilon_2)$-Nash equilibrium.*

Proof.

$$\begin{aligned}
u_1(\overline{\sigma}_1^T, \overline{\sigma}_2^T) &= \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, \overline{\sigma}_2^T) && (2.4)\\
&\geq \frac{1}{T} \min_{\sigma_2^* \in \Sigma_2} \sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_2^*) && (2.5)\\
&\geq \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_2^t) - \varepsilon_2 && (2.6)\\
&\geq \frac{1}{T} \max_{\sigma_1^* \in \Sigma_1} \sum_{t=1}^{T} u_1(\sigma_1^*, \sigma_2^t) - \varepsilon_1 - \varepsilon_2 && (2.7)\\
&\geq u_1(\sigma_1', \overline{\sigma}_2^T) - \varepsilon_1 - \varepsilon_2 \qquad \forall \sigma_1' \in \Sigma_1 && (2.8)
\end{aligned}$$

Steps 2.4 and 2.8 use the definition of average strategy and the linearity of expectation. Step 2.6 uses the regret bound for player 2 and step 2.7 uses the regret bound for player 1. An analagous proof can used to show

$$u_2(\overline{\sigma}_1^T, \overline{\sigma}_2^T) \geq u_2(\overline{\sigma}_1^T, \sigma_2') - \varepsilon_1 - \varepsilon_2 \qquad \forall \sigma_2' \in \Sigma_2 \tag{2.9}$$

which completes the proof by Definition 6. □

Efficient algorithms exist for minimizing an agent's regret in the **adversarial multi-armed bandit** setting, where at each time step the agent selects an action from a set A and the adversary (without knowing the agent's choice) assigns payoffs to each element in A. This can be considered equivalent to an extensive-form game where each player has only one information set, and the agent's strategies are exactly the elemets of A. Algorithms for minimizing regret in this setting include **(linear) regret-matching** [20], which is derived from Blackwell's Approachability Theorem [5], and **Hedge** [10], which is a generalized form of the Weighted Majority Algorithm [30]. These approaches can be united under a regret-matching framework [19].

Regret minimization algorithms can easily be extended to extensive-form games by considering the set of pure strategies as the set of actions. Unfortunately, when we do so, we lose the efficiency these algorithms had in the bandit setting, as their time and space complexity will now depend on the size of the strategy set, which is exponential in the size of the game. The key insight that allows efficient regret minimization in extensive-form games is that a player's overall regret can be decomposed into individual values at each information set, allowing overall regret to be minimized by independently minimizing a form of regret at every information set.

2.4.1 Counterfactual Regret Minimization

Counterfactual regret minimization (CFR) is an algorithm developed by Zinkevich et al. [36] for minimizing regret in extensive form games. The algorithm works by independently minimizing a player's **counterfactual regret** at every information set. Counterfactual regret is defined using **counterfactual value**, which is a player's expected utility for reaching a given information set, assuming that player modifies his strategy to reach the information set when possible.

Definition 8. [29] *The* **counterfactual value** *(or* **counterfactual utility***) for player i under strategy profile σ at information set I is*

$$v_i(\sigma, I) = \sum_{z \in Z_I} \pi^{\sigma}_{-i}(z[I])\pi^{\sigma}(z[I], z)u_i(z) \tag{2.10}$$

where $Z_I = \{z \in Z : \exists h \in I, h \sqsubseteq z\}$ is the set of terminal histories that go through I and $z[I] \in I$ is the unique history in I such that $z[I] \sqsubseteq z$.

Definition 9. [26] *Consider a repeated extensive form game in which strategy profile σ^t is played at time t. The* **counterfactual regret** *of player i at time t for action a at information set I is*

$$r^t_i(I, a) = v_i(\sigma^t_{(I \to a)}, I) - v_i(\sigma^t, I) \tag{2.11}$$

where $\sigma^t_{(I\to a)}$ is the strategy profile σ^t except at information set I, action a is always played.

A player's overall regret in a extensive-form game is equal to a summation over counterfactual regrets. For any x, define $x^+ = \max\{x, 0\}$.

Theorem 3. [16, 36]

$$R^T_i = \sum_{I \in \mathcal{I}_i} \pi^{\sigma^*}_i(I) \sum_{a \in A(I)} \sigma^*_i(a|I) \frac{1}{T} \sum_{t=1}^{T} r^t_i(I, a) \leq \sum_{I \in \mathcal{I}_i} \max_{a \in A(I)} \left(\frac{1}{T} \sum_{t=1}^{T} r^t_i(I, a) \right)^+ \tag{2.12}$$

where $\sigma^*_i = \arg\max_{\sigma'_i \in \Sigma_i} \sum_{t=1}^{T} u_i(\sigma'_i, \sigma^t_{-i})$.

This theorem shows that if we independently minimize the counterfactual regrets at each information set, then we minimize the average overall regret, and the average strategy profile will converge to a Nash equilibrium. CFR creates an iterated series of strategy profiles $\sigma^1, \sigma^2, ...$ by using the linear regret-matching update rule (with counterfactual regrets) to update $\sigma^t_i(\cdot|I)$ at each information set [36]. This update rule is:

$$\sigma^{T+1}_i(a|I) = \begin{cases} \frac{\sum_{t=1}^{T} r^t_i(I,a)}{\sum_{b \in A(I)} \sum_{t=1}^{T} r^t_i(I,b)} & \text{if } \sum_{b \in A(I)} \sum_{t=1}^{T} r^t_i(I, b) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases} \tag{2.13}$$

Thus CFR assigns each action a probability proportional to accumulated counterfactual regret from not playing that action. The values are updated by walking the tree from the root.

We can use the linear regret-matching bound given by Gordon [18] in combination with Theorem 3 to bound the average overall regret of the CFR algorithm after T iterations. Define $\sigma_i^* = \arg\max_{\sigma'_i \in \Sigma_i} \sum_{t=1}^{T} u_i(\sigma'_i, \sigma_{-i}^t)$. For each player i, let $\mathcal{B}_i$ be the partition of $\mathcal{I}_i$ such that two information sets I, I' are in the same $B \in \mathcal{B}_i$ if and only if player i takes the same sequence of actions to reach I and I'. Define the **M-value of** σ_i to be $M_i(\sigma_i) = \sum_{B \in \mathcal{B}_i} \sqrt{|B|} \max_{I \in B} \pi_i^{\sigma_i}(I)$. Let $\Delta_i = \max_{z,z' \in Z} u_i(z) - u_i(z')$ be the range of utilities for player i. Let $|A_i| = \max_{h \in H: P(h)=i} |A(h)|$ be player i's maximum number of available actions.

Theorem 4. [16] *When using the regret matching update rule in CFR, the average regret after T iterations is bounded as*

$$R_i^T \leq \frac{\Delta_i M_i(\sigma_i^*) \sqrt{|A_i|}}{\sqrt{T}} \tag{2.14}$$

Because the regret after T iterations is $\mathcal{O}(1/\sqrt{T})$, we know from Theorem 2 that CFR takes $\mathcal{O}(1/\varepsilon^2)$ iterations to compute a ε-Nash equilibrium. Also worth noting is that $M_i(\sigma_i^*) \leq |\mathcal{I}_i|$, so the bound is linear in the number of information sets. In addition, the algorithm only needs to store regrets and strategies for information sets, so the memory usage is linear in the number of information sets.

2.4.2 Sampling in CFR

The traditional CFR algorithm walks the entire game tree, and updates the regrets at each information set on every iteration. In practice, faster convergence is seen when some sort of sampling is used to select which part of the tree to walk on a particular iteration. For example, the original paper of Zinkevich et al. proposed using **chance sampling**, in which one action is sampled for each chance node on every iteration. Only information sets consistent with the sampled chance events are updated on that iteration [36].

In this thesis, we will make particular use of another sampling scheme called **public chance sampling (PCS)** [26]. Under PCS, chance events are sampled only if the outcome is immediately visible to all players (meaning that two histories with different outcomes for the chance event can never be in the same information set). Instead of explicitly walking each private chance outcome when a chance node is encountered in the tree, the algorithm defers the evaluation of private chance events until a terminal node is reached. Under a normal tree walk, if each player has n private chance outcomes, $\Theta(n^2)$ terminal evaluations must be performed. The public tree walk used in PCS exploits the payoff structure of some games (such as poker games where payoffs depend on a ranking of private chance outcomes) to speed this up to $\Theta(n)$ evaluations (possibly with a $\Theta(n \ln n)$ sorting step).

2.5 Abstraction

Even with the unprecedented scaling offered by CFR, there are many interesting extensive-form games that are too large to solve with existing computing resources. Two-player limit Texas Hold'em poker, the smallest poker game which is regularly played by humans, has 10^{18} game states and 10^{14} information sets [34], which means that traditional CFR would need over a petabyte of RAM to solve the game, with time constraints being similarly limiting. [1]

When working with games that are too large to solve directly with CFR, the majority of research uses a method of **abstraction** to try to find a good strategy for the game. Abstraction creates a smaller extensive form game which is meant to approximate the full game. An equilibrium finding technique such as CFR is used to find an optimal strategy for the abstract game. This strategy is then translated back to the full game. It is hoped that the abstract game equilibrium will be close to a full game equilibrium.

There are two commonly used methods for creating an abstract game from a full game in a structured fashion. The first uses the full game's underlying

[1] Recently, Bowling et al. were able to use a modified form of CFR to find an ε-Nash equilibrium for two-player limit Texas Hold'em, but their technique used over 900 core-years of computation time and 10.9 TB of disk space [6].

game tree for the abstract game, but shrinks the game's information tree by merging player information sets to create a coarser information partition. The second shrinks the game tree itself by removing legal actions from the abstract game. Waugh et al. formally defined abstraction that uses either or both of these techniques:

Definition 10. [33] *An* **abstraction for player** *i is a pair $\alpha_i = \langle \alpha_i^{\mathcal{I}}, \alpha_i^{A} \rangle$, where*

- *$\alpha_i^{\mathcal{I}}$ is a partitioning of H_i, defining a set of abstract information sets. This partition must be coarser than $\mathcal{I}_i$: if $h, h' \in I$ for some $I \in \mathcal{I}_i$ then $h, h' \in I'$ for some $I' \in \alpha_i^{\mathcal{I}}$.*
- *α_i^{A} is function from histories to sets of actions, where $\alpha_i^{A}(h) \subseteq A(h)$ and $\alpha_i^{A}(h) = \alpha_i^{A}(h')$ for all $h, h' \in H$ such that $h \in I$ and $h' \in I$ for some $I \in \alpha_i^{\mathcal{I}}$.*

An **abstraction** *α is a set of α_i, one for each player. The* **abstract game** *Γ^{α} is the extensive form game obtained from Γ by replacing $\mathcal{I}_i$ with $\alpha_i^{\mathcal{I}}$ and $A(h)$ with $\alpha_i^{A}(h)$ for all $h \in H_i$.*

In general, we will assume that the abstraction maintains perfect recall in the abstract game.

In limit Texas Hold'em, chance events have a very large number of possible outcomes (and thus branching factor), while player nodes have a relatively small action set. Because of this, limit Texas Hold'em abstraction is focused on merging information sets by making certain chance outcomes indistinguishable to the players. The chance actions in limit Texas Hold'em are the deal of the cards, so by mergining information sets previously differentiated by chance actions, we are causing players to be unable to differentiate hands (collections of cards including private cards and community cards) that they could differentiate in the full game. We will refer to this method of abstraction as **card abstraction**. Conveniently, a strategy in such an abstract game can be directly played in the full game (because every full game information set is a subset of some abstract game information set).

This thesis is not concerned with how abstractions are created, but only with how they are used. For examples of modern abstraction techniques, see [34, 27, 13, 7].

2.5.1 Solution Concepts in Abstractions

The simplest way to find a solution strategy in a abstract game is to use a technique such as CFR to find an ε-Nash equilibrium for the abstract game. Because the abstraction is designed to be similar to the full game, it is hoped that the the abstract equilibrium will be similar to the full game equilibrium. However, there is no guarantee on the quality of an abstract game equilibrium when played in the full game. Waugh et al. found that abstract game equilibria for Leduc Hold'em can have widely varying exploitabilities in the full game, and the sizes of these exploitabilities are not well-correlated with how closely the abstraction approximates the full game [33].

One proposed alternative to finding an abstract game equilibrium is to find an abstract game strategy which has minimal exploitability in the full game. This can be done with the technique of **CFR-BR**, which runs CFR updates for one player, but replaces the other CFR player with a best-response (BR) player [25]. On each iteration of the algorithm, the BR player is updated to play a full game best-response to the CFR player's strategy (which is defined in an abstract game). This guarantees that the CFR player's strategy will converge to a strategy that minimizes full game exploitability amongst the strategies that are expressible in the abstract game.

The use of CFR-BR is limited in practice because it requires an efficient method for computing a best-response in the full game. The output of the CFR-BR computation is equivalent to solving for an equilibrium in an abstraction where one player (the CFR player) is abstracted and the other (the BR player) plays in the full game. The advantage of CFR-BR is that it saves on memory by not needing to store the regrets of the BR player, but it still requires comparable computation time to when CFR is run with an unabstracted player [8].

2.6 Responding to an Opponent Model

Many extensive-form games, including poker games, have more than a simple binary outcome of winning or losing. Also important is the range of utility outcomes; a zero-sum game is guaranteed to have one winner and one loser, but the winner could win by many different amounts. While equilibrium strategies are guaranteed to not lose against any opponent, they might not win as much as other strategies could against exploitable opponents. This makes another interesting research question for extensive form games: how can we create agents that take advantage of exploitable opponents without giving up too much?

Some research has looked at learning to take advantage of an opponent in an online setting (for examples, see [11, 3]), but in very large games such as poker games, it can be hard to learn an accurate opponent model quickly. The majority of research has thus focused on offline techniques. These techniques begin by creating a model of the opponent's strategy (perhaps using action records from previous play), and then learning a good counter-strategy. The naive technique would be to simply learn a best-response strategy to the model, but this is a poor idea in practice; a best-response strategy is very brittle, meaning that if the opponent model is even only slightly inaccurate, the best-response strategy can perform poorly against the actual strategy.

The technique of **Restricted-Nash Response (RNR)** was developed to solve this problem [23]. RNR uses an input parameter p with $0 \leq p \leq 1$, called the **mixing rate**. It then runs a modified form of the CFR algorithm, where one player is updated as usual, but the other player is replaced with what is called a **RNR mimic**. The mimic does the normal CFR update with probability $1 - p$, but with probability p it plays the static strategy that we are using to model the opponent. The CFR player's strategy is the output. At $p = 0$, the RNR mimic is a normal CFR player, and thus the RNR strategy will be an equilibrium strategy. At $p = 1$, the RNR mimic always plays the opponent model, and the RNR strategy will be a best-response. Using a p between 0 and 1 allows the RNR strategy to exploit the opponent while giving up comparatively little of its own exploitability.

It can be proved that RNR produces strategies that are optimal at exploiting an opponent while limiting the strategy's own exploitability.

Theorem 5. [23] *For any $\sigma_{-i} \in \Sigma_{-i}$, let σ_i be the strategy produced by running RNR with σ_{-i} as the opponent model. Then there exists some ε such that*

$$\sigma_i = \operatorname*{arg\,max}_{\sigma_i^* \in \Sigma_i^{\varepsilon-\text{safe}}} u_i(\sigma_{-i}, \sigma_i^*)$$

Where $\Sigma_i^{\varepsilon-\text{safe}} = \{\sigma_i \in \Sigma_i \colon \text{exploit}(\sigma_i) \leq \varepsilon\}$ is the set of strategies exploitable for no more than ε.

Chapter 3

Strategy Evaluation

In Section 2.3, we presented Nash equilibria as a solution concept for extensive-form games. In games which are too large for a Nash equilibrium to be easily computed, various techniques are used in an attempt to find strategies that approximate a Nash equilibrium; these techniques include running CFR (Section 2.4.1) or CFR-BR (Section 2.5.1) in an abstract game (Section 2.5). In order to compare these techniques, we would like to evaluate the strategies that they produce.

The natural way to measure which of two strategies is stronger is to compare them directly in head-to-head play. However, this method is hampered by the fact that expected utility between strategies is not transitive. It is common for σ^1 to defeat (in expectation) σ^2 and σ^2 to defeat σ^3, and yet still have that σ^3 defeats σ^1 in expectation. Perhaps the issue can be avoided by comparing performance against a variety of opponents, but this introduces additional difficulties such as how the set of opponents is chosen and how performance against each one is weighted; is it better to defeat one opponent by a lot and lose to many by a small amount, or is it better to have uniformly small wins?

In this chapter we will discuss various ways that strategies can be evaluated. In section 3.1 we examine exploitability, a commonly used metric in previous work, and discuss its limitations. In section 3.2 we look at other evaluation techniques that have been suggested in previous work. In section 3.3 we introduce a new framework that can encompass both pre-existing and new evaluation methods and present some examples. In section 3.4 we give an example that shows how our framework can give an enhanced conception of strategy strength.

3.1 Using Exploitability to Evaluate Strategies

As we introduced in section 2.3.1, exploitability is a measure of how much a strategy (or profile) loses against a worst-case opponent. Exploitability is closely connected to the concept of ε-Nash equilbria; if $\text{exploit}(\sigma) \leq \frac{\varepsilon}{2}$ then σ is an ε-Nash equilibrium. Thus as the exploitability of σ decreases, it becomes a closer and closer approximation of a Nash equilibrium. Because of this connection, exploitability is a natural metric for evaluating strategies when we use Nash equilibria as our solution concept.

Computing exploitability requires computing a best-response, which is not possible in all domains. While best-response computation is simpler than equilibrium computation, it is still intractable in many large games. Only recent algorithmic advances have allowed the computation of best-responses in two-player limit Texas Hold'em, and computation still takes 76 CPU-days [24]. In larger games, we cannot compute a best-response, so we cannot use exploitability to evaluate strategies.

Even in games where the exploitability calculation is tractable, its use is limited. Using exploitability as a strategy evaluation metric was the inspiration behind the CFR-BR algorithm (see section 2.5.1), as it finds a strategy that minimizes exploitability (in the full game). However, strategies produced by CFR-BR seem to perform poorly in practice. Not only does a CFR-BR strategy lose to an abstract game Nash equilibrium from the same abstraction in one-on-one play, the Nash equilibrium also loses less than the CFR-BR strategy does when each is played against a Nash equilibrium from a larger abstraction [25].

The issues with CFR-BR are just part of a series of results which indicate that exploitability is not a good predictor of one-on-one performance. Waugh performed an experiment comparing abstract game equilibria from various

Leduc Hold'em abstractions [32]. Each pair of strategies was played against each other to determine the expected utility for each strategy, and then the strategies are ranked in two fashions. In the **total bankroll** ranking, strategies are ordered by their average expected utility against all other strategies. In the **instant runoff** setting, the strategy with the worst expected utility is iteratively removed until one winner remains. Waugh found a correlation coefficient between exploitability and ranking of only 0.15 for total bankroll and 0.30 for instant runoff [32]. Furthermore, Johanson et al. evaluated several of the strategies submitted to the limit Texas Hold'em event at the 2010 Annual Computer Poker Competition. The winner of the competition by instant runoff ranking was more exploitable than three of the strategies it defeated, and tended to have better performance than these less exploitable strategies when playing against the other agents [24]. Bard et al. created abstract game equilibria for a variety of Texas Hold'em abstractions and found that when the size of an abstract game is varied for only one player's information sets, exploitability and one-on-one performance against the other equilibria were inversely correlated [4].

These results can likely be explained by the fact that exploitability is a *pessimistic* metric (previously discussed in [14]). Exploitability measures the worst-case performance of a strategy, but not how common it is for the worst case to be realized. For example, the **always-raise** strategy which always makes the bet/raise action is almost five times as exploitable as the **always-fold** strategy [24], but always-fold achieves its worst case against any strategy that doesn't immediately fold itself, while always-raise achieves its worst case only against an opponent that makes a specific series of actions for each hand.

Even if an opponent is actively trying to exploit our strategy, exploitability can be a poor metric. In the large games we are most concerned with, computing a best-response is likely to be too resource-intensive for the opponent to do online, even if he has access to our full strategy. If he doesn't have access to our strategy and must learn it during game play, the situation is even more bleak. Thus, as long as our strategy is private, it is not clear that we need to worry about opponents learning to maximally exploit it.

When we consider the pessimistic nature of exploitability, we can formulate a hypothesis to explain why CFR-BR strategies tend to be weak. We should note that the CFR-BR algorithm uses a form of exploitability that is even *more* pessimistic than usual. The CFR player is abstracted, while the BR player is not, so the CFR player isn't just assuming that her opponent knows how to best respond to her, but she is also assuming that her opponent has more information than she does. Thus the CFR-BR strategy will learn to be overly cautious, and will avoid taking actions that might have positive expected value against many opponents in favor of actions that minimize losses against all opponents.

3.2 Other Evaluation Techniques

The problems with exploitability have not gone unnoticed in previous work. Some other evaluation techniques have been proposed to supplement the information that exploitability gives. Most of these suggestions involve evaluating a strategy in one-on-one play, either against a particular opponent or against a group of opponents.

In small games, for example, abstract game equilibria have been evaluated by playing them against a full game equilibrium [17, 14]. Of course, this is not practical in most domains, since abstraction is used when finding a full game equilibrium is intractable. It also isn't clear how to choose the equilibrium in games with multiple equilibria.

Another common approach is for strategies to be played against a set of benchmark opponents, with the results presented in a **crosstable** that shows the outcome between every pair of strategies. Opponents have often been taken from past competitors in the Annual Computer Poker Competition [36, 14, 12, 3]. In other cases, researchers have tested an algorithm by running it multiple times with different parameters[1], and then put the resulting strategies in a crosstable, possibly with a reference strategy [34, 4, 35]. The results of a crosstable can be interpreted in numerous ways, but often strategies are

[1] For example, by running it in different abstract games.

compared by looking at their performance averaged across all opponents.

Waugh proposed a new metric specifically designed to replace exploitability when predicting one-on-one performance against equilibrium-like (i.e. abstract game equilibria) strategies. This metric, called **domination value**, measures the performance of a strategy against a worst-case equilibrium opponent.

Definition 11. [32] *The* **domination value** *of a strategy $\sigma_i \in \Sigma_i$ is*

$$\mathrm{Dom}_i(\sigma_i) = \max_{\sigma_{-i} \in \Sigma^*_{-i}} u_{-i}(\sigma_i, \sigma_{-i}) - v_{-i} \tag{3.1}$$

where Σ^*_{-i} is the subset of Σ_{-i} containing strategies which are part of some Nash equilibrium for the game Γ.

Domination value is thus analogous to exploitability, but with the opponent limited to only playing (full game) Nash equilibrium strategies. The intuition here is that a strategy with a low domination value will play fewer dominated actions, and thus will make fewer mistakes that lose money against any strong opponent, even one that doesn't learn. Results showed that domination value was better than exploitability at predicting one-on-one performance, at least against abstract game equilibria. Using the same Leduc Hold'em abstract equilibria described in the previous section, Waugh found that domination value had a correlation coefficient of 0.84 with bankroll results, and 0.87 with runoff results [32]. Unfortunately, computing a strategy's domination value is not practical in the large games that we are interested in, as it requires solving a linear program with size proportional to the size of the game, and is thus at least as hard as finding a (full game) equilibrium.

Finally, when we specifically evaluate abstract game equilibria, we can also use techniques designed to evaluate the abstraction directly. One such technique is to run CFR-BR in the abstraction and find the exploitability of the resulting strategy. Johanson et al. found that the exploitability of the CFR-BR strategy was better than the exploitability of the equilibrium strategy at predicting the one-on-one performance of the equilibrium strategy [27]. In this thesis, we are interested in techniques that can evaluate strategies in general, as opposed to only evaluating abstract game equilibria, so this result is of limited use.

3.3 A Generalized Evaluation Framework: Pretty-Good Responses

So far, we have considered a variety of techniques for evaluating strategies. The majority of these techniques take a common form. Given an input strategy to be evaluated, some opponent strategy (or strategies) is chosen. For one-on-one metrics, this is some predetermined static strategy, for exploitability or domination value, it is a function of the input strategy. The input strategy is then played against the response strategy, and is evaluated based on its expected utility. We formalize this general technique by introducing notation for **response functions** and **response value functions**.

A **response function** is a mapping from the strategy space of one player to the strategy space of the other player. Throughout the rest of this thesis, we will often assume without loss of generality that we are evaluating strategies for player 1, with player 2 designated as the response player. Then a response function is any function $f\colon \Sigma_1 \to \Sigma_2$. A **response value function** is a real-valued function that gives the expected utility of a strategy when it is played against the output of a response function. We write the response value function induced by a response function f as v_f, and it is defined as $v_f = u_1(\sigma_1, f(\sigma_1))$. Under this framework, exploitability is the response value function induced by the best-response function (though the value is negated by convention), and one-on-one performance against an opponent σ_2 is the response value function induced by the response function that returns σ_2 for all inputs.

With the notation laid out, we now turn to the problem of creating a metric that accounts for the deficiencies of exploitability. Above, we argued that exploitability fails at predicting one-on-one performance because it gives the opponent too much credit in regards to his ability to learn and respond to our strategy. The other extreme is to assume that the opponent has no ability to learn and respond to our strategy; instead, we require him to play a static strategy that doesn't respond to our strategy. This has its own issues, though, as our best strategy under such a metric is a best-response to the opponent strategy, but as we discussed in section 2.6, a best-response strategy is very brittle. That means that a one-on-one evaluation technique is sensitive to even small changes in our opponent strategy of choice, and can be a poor predictor of one-on-one performance against opponent strategies that differ from the evaluation strategy.

Instead, we would like an evaluation metric that is somewhere in between; it should try to exploit the strategy being evaluated, but it shouldn't be able to do so with the perfect exploitive ability of a best-response. We thus introduce a family of exploitative response functions that can be varied between no exploitative strength (a static opponent strategy) and perfect exploitative strength (a best-response).

Definition 12. *A function $f\colon \Sigma_{-i} \to \Sigma_i$ is called a* **pretty-good response (PGR)** *if $u_i(\sigma_{-i}, f(\sigma_{-i})) \geq u_i(\sigma_{-i}, f(\sigma'_{-i}))$ for all $\sigma_{-i}, \sigma'_{-i} \in \Sigma_{-i}$.*[2]

The pretty-good response condition requires that the responder maximizes his utility when he responds to the correct opponent strategy; he can't increase his utility by responding to σ' (and thus playing $f(\sigma')$) when the opponent is actually playing σ. This ensures that the responder is making an attempt to exploit his opponent. To illustrate this, consider the modeling setting, in which the response player in some way builds a model of his opponent's strategy (e.g. through online or offline sampling), and then applies a response function to the model to choose his own strategy. If he uses a response function that is not a pretty-good response, it can result in a situation where the responder can exploit the opponent more if he has an inaccurate model for his opponent strategy. Thus, pretty-good response functions can be viewed as the class of functions that "correctly" exploit a model.

We can see the flexibility of the pretty-good response framework by establishing a connection to subsets of responder strategies.

Lemma 1. *For every $f\colon \Sigma_{-i} \to \Sigma_i$ such that f is a pretty-good response, there is a corresponding $\Sigma_i^f \subseteq \Sigma_i$ such that*

$$f(\sigma_{-i}) = \arg\max_{\sigma_i^* \in \Sigma_i^f} u_i(\sigma_{-i}, \sigma_i^*)$$

[2] We use i and $-i$ here because PGRs are defined even in games with more than two players. In our WLOG framework for two-player games, $i = 2$ and $-i = 1$.

Similarly, for every $\Sigma_i^f \subseteq \Sigma_i$, there is a corresponding $f\colon \Sigma_{-i} \to \Sigma_i$ such that $f(\sigma_{-i}) \in \Sigma_i^f$ for all σ_{-i} and f is a pretty-good response.

Proof. For the forward direction, let $\Sigma_i^f = \{f(\sigma_{-i})\colon \sigma_{-i} \in \Sigma_{-i}\}$. For the reverse direction, define $f(\sigma) = \arg\max_{\sigma_i^* \in \Sigma_i^f} u_i(\sigma_{-i}, \sigma_i^*)$. □

By defining Σ_i^f to be a singleton $\{\sigma_i\}$, we get an f that always generates σ_i, and thus a v_f that measures one-on-one performance. By defining $\Sigma_i^f = \Sigma_i$, we get an f that produces a best-response, and thus a v_f that measures exploitability. By choosing any Σ_i^f that contains more than one strategy, but not all strategies, we can get an f that is exploitative (i.e. it is a response function that optimizes its utility based on the opponent's strategy) but has only partial exploitative power (i.e. it can't generate a perfect best-response to all opponents).

Other evaluation metrics also fit under the pretty-good response framework. Domination value is generated by choosing a Σ_i^f which contains strategies from Σ_i if and only if they are part of some Nash equilibrium. Worst-case performance in a crosstable is generated by choosing a Σ_i^f corresponding to the other strategies in the crosstable. Average performance in a crosstable is generated by choosing a Σ_i^f corresponding to the singleton strategy that is an average of all other crosstable strategies.[3] Exploitability in an abstract game α is generated by choosing $\Sigma_i^f = \Sigma_i^\alpha$, which is the set of strategies for player i that are **consistent** with α.[4]

3.4 An Example Showing the Power of Pretty-Good Responses

One of the motivating examples for developing the PGR framework was the difference between CFR and CFR-BR strategies. CFR-BR strategies are both theoretically and empirically less exploitable than CFR strategies, but CFR

[3] Pretty-good response equivalence might fail for crosstables that contain players which play something other than a static strategy.

[4] A strategy is consistent with an abstraction if it assigns zero probability to any actions that don't occur in the abstraction, and it assigns identical probability distributions to information sets which are merged in the abstraction.

strategies seem to be “stronger” in practice, when we measure their strength in one-on-one play. In this section, we evaluate a CFR-BR and a CFR strategy, each created in an abstracted version of Texas Hold’em. When these strategies are played against exploitative opponents with perfect exploitative power (i.e. a best-response), we know that CFR-BR will come out ahead because of its lower exploitability. Instead, we examine how the strategies fair against exploitative opponents that have only partial exploitative power. We do this by using the PGR framework.

In particular, we use abstract game best-responses as our pretty-good responses. As discussed in the previous section, an abstract game best-response is a PGR because it maximizes the responder’s utility among a specific subset of the responder’s strategies (i.e. the strategies consistent with the abstraction). We construct a series of abstract games from the full game of Texas Hold’em, with each abstraction having a different size, and then compute best-responses in these games.

We know that a CFR strategy is an abstract game equilibria, and thus is guaranteed to be the optimal strategy when evaluated by an abstract game best-response in the same abstraction. It’s reasonable to expect that CFR will also be advantaged over CFR-BR in abstractions that are very similar to the training abstraction. To minimize this effect when evaluating the strategies, we used different abstraction techniques when creating the training abstraction and when creating the evaluation abstraction. For training the CFR and CFR-BR strategies, we used percentile bucketing with a combination of **expected hand strength** ($\mathbb{E}[HS]$) and **expected hand strength squared** (($\mathbb{E}[HS^2]$), where **hand strength** is the percentage of possible opponent hands that a given hand beats after the river has been dealt. For the evaluation abstractions, we used k-means clustering with histogram-based distance measures as described in [27].

The results can be seen in Figure 3.1. The two lines respectively represent a single strategy produced by CFR and a single strategy produce by CFR-BR. Each point on a line shows the amount of expected utility that the strategy loses against an abstract game best-response; a smaller value thus corresponds

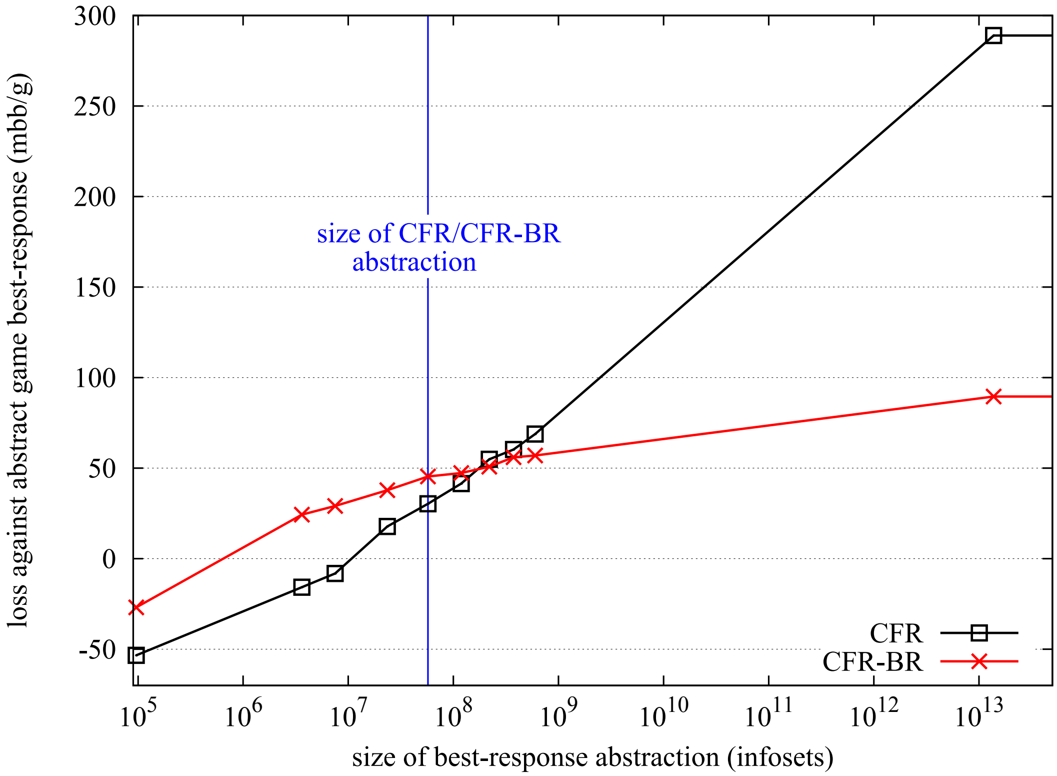

Figure 3.1: Comparison of a CFR strategy and a CFR-BR strategy using various abstract game best-responses.

to a better performance for the CFR or CFR-BR strategy. The values are given in **milli-big-blinds per game (mbb/g)**, which is the expected utility, normalized with respect to the size of the big blind's forced bet and multiplied by 1000. The x-axis shows the size of the abstract game used when creating the abstract game best-response. The rightmost point on each line was calculated in a lossless abstraction[5], and thus shows the exploitability of the strategy. The vertical line shows where the abstract game best-responses were calculated in an abstraction of the same size as the one used to train the CFR and CFR-BR strategies.

As the far right of this graph shows, CFR-BR is much less exploitable than CFR. By comparing the lines on the left side of the graph, however, we see that the CFR-BR strategy is comparatively easier to exploit with sim-

[5] The lossless abstraction treats two hands as equivalent if they are isomorphic under a permutation of card suits.

pler strategies. Even in an abstraction approximately twice as large as the training abstraction, the CFR-BR strategy loses more to its abstract game best-response than the CFR strategy does, and this difference becomes bigger as the response abstraction gets smaller. Although the CFR-BR strategy loses less in its absolute worse case, it loses amounts which are comparable to this worst case against much weaker opponents. In contrast, the CFR strategy has relatively poor worst-case performance, but only very complex opponents can closely approximate the worst case.

When we use exploitability to evaluate these strategies, we are only looking at the narrow slice on the far right of this graph. However, any particular vertical slce of the graph can have predictive power against the right set of opponents. Even when we give our opponent total access to our strategy and allow him to exploit us, relative performance can differ from relative exploitability when the opponent's computational power is limited. The graph shows us that a CFR strategy is harder for our opponent to exploit if we assume that he is limited to using an abstraction that is of comparable size to our own. Pretty-good responses can thus help us to see a fuller picture of strategy strength than we can get from exploitability alone.

Chapter 4

Using Pretty Good Responses to Train Strategies

In the previous chapter we showed that pretty-good responses can be useful for evaluating strategies. In particular, in Section 3.4 we used a variety of pretty-good responses (PGR) to compare the strategies computed by CFR and CFR-BR. These techniques were each designed to find an optimal strategy given a particular evaluation metric; CFR finds the strategy with optimal exploitability[1] and CFR-BR finds the strategy with optimal full game exploitability. If we are going to use a PGR to evaluate strategies, then when we are training a strategy in the first place, we should ideally use a technique that optimizes with respect to the PGR.

In this chapter, we examine how PGRs can be incorporated into strategy computation techniques. In Section 4.1 we look at how any regret minimization algorithm can be used to find a strategy optimal under PGR evaluation. In Section 4.2 we look at how the theoretical bounds change for response functions that only approximate the PGR condition. In Section 4.3 we show how the Restricted Nash Response technique can be generalized with PGRs.

[1] If CFR is run in an abstract game as it was in our example, it finds the strategy with optimal abstract game exploitability.

4.1 Regret Minimization with Pretty-Good Responses

In the CFR-BR algorithm, a regret minimization algorithm (i.e. CFR) is run for one player's strategy, while the other player is constrained to playing the strategy specified by a response function (i.e. best-response). Similarly, we can run any regret minimization algorithm to choose a series of strategies $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T$ for player 1 while using any response function f to fix player two's strategy to be $f(\sigma_1^t)$ on each iteration t. When we do this with CFR as our regret minimization algorithm, we call the resulting algorithm CFR-f.

To analyze the results of running CFR-f, we introduce the notion of **no-regret learnable** response functions.

Definition 13. *Let $f\colon \Sigma_1 \to \Sigma_2$ be a response function and define $\sigma_2^t = f(\sigma_1^t)$ for $t = 1, ..., T$. f is called* **no-regret learnable** *if for every sequence of strategies $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ such that $R_1^T \leq \varepsilon$ (where $\varepsilon > 0$), we have that*

$$u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + \varepsilon \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, f(\sigma_1^*))$$

where $\overline{\sigma}_1^T$ is the mixed strategy that mixes equally between each of $\sigma_1^1, ..., \sigma_1^T$.

The notion of no-regret learnable is important because many regret minimization algorithms (i.e. CFR, adversarial bandit algorithms) are guaranteed to achieve subconstant regret R_1^T against any opponent. Thus by running CFR-f using a no-regret learnable f, the average strategy for the CFR player is guaranteed to converge to a strategy that maximizes the corresponding response value function v_f. Although we use CFR-f as an example throughout this chapter, it's important to note that the theory applies if we replace CFR with any regret minimization algorithm.

We show that all PGRs are no-regret learnable.

Theorem 6. *Every $f\colon \Sigma_1 \to \Sigma_2$ that is a pretty-good response is no-regret learnable.*

Proof. Let $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ be a sequence such such that $R_1^T \leq \varepsilon$ (defined using $\sigma_2^t = f(\sigma_1^t)$).

$$\begin{aligned}
\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) &= \frac{1}{T} \max_{\sigma_1^* \in \Sigma_1} \sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^*)) && (4.1)\\
&\leq \frac{1}{T} \max_{\sigma_1^* \in \Sigma_1} \sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^t)) && (4.2)\\
&\leq \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) + \varepsilon && (4.3)\\
&\leq \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, f(\overline{\sigma}_1^T)) + \varepsilon && (4.4)\\
&= u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + \varepsilon && (4.5)
\end{aligned}$$

Steps 4.1 and 4.5 use the definition of average strategy and the linearity of expectation. Steps 4.2 and 4.4 use the PGR condition. Step 4.3 uses the fact that $R_1^T \leq \varepsilon$. □

Thus for any PGR f, the average strategy generated by CFR-f is guaranteed to converge to a strategy that optimizes the evaluation metric that uses f.

The fact that PGRs are no-regret learnable guarantees only that the average strategy of the CFR player in CFR-f converges. We now show that the current strategies $(\sigma_1^1, ..., \sigma_1^T)$ generated by CFR-f converge with high probability if f is a PGR.

Theorem 7. *Let $f \colon \Sigma_1 \to \Sigma_2$ be a pretty-good response and $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ be a sequence of strategies such that $R_1^T \leq \varepsilon$ when $\sigma_2^t = f(\sigma_1^t)$ for $t = 1, ..., T$. Then if we select T^* uniformly at random from $[1, T]$, we get that*

$$u_1(\overline{\sigma}_1^{T^*}, f(\overline{\sigma}_1^{T^*})) \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon}{p}$$

with probability at least $1 - p$ for any $p \in (0, 1]$.

Proof. From (4.3) in the proof of Theorem 6, we have that

$$\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) \leq \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) + \varepsilon \qquad (4.6)$$

From the definition of maximum, we also have that

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) &\leq \frac{1}{T}\sum_{t=1}^{T} \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) && (4.7)\\ &= \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) && (4.8)\end{aligned}$$

Let k be the number of indices $t \in \{1, ..., T\}$ such that $u_1(\sigma_1^t, f(\sigma_1^t)) < \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon}{p}$. Then

$$\begin{aligned}&\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \varepsilon \leq \frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) && (4.9)\\ &\quad < \frac{1}{T}\left(k\left(\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon}{p}\right) + (T-k)\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*))\right) && (4.10)\\ &\quad = \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon k}{pT} && (4.11)\\ &\therefore k < pT && (4.12)\end{aligned}$$

The left part of step 4.10 comes from the definition of k and the right part comes from (4.8). We thus have that there are at most pT strategies in $\{\sigma_1^1, ..., \sigma_1^T\}$ such that $u_1(\sigma_1^t, f(\sigma_1^t)) < \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon}{p}$, so if T^* is selected uniformly at random from $\{1, ..., T\}$ there is at least a $1-p$ probability that $u_1(\sigma_1^{T^*}, f(\sigma_1^{T^*})) \geq \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon}{p}$. □

Thus as long as CFR-f is run so as to stop stochastically, the final strategy will be near optimal with high probability. In practical terms, this means that a CFR-f implementation can cut memory usage in half because it doesn't need to track the average strategy.

4.2 Response Functions that Deviate from Pretty-Good

We argued in Chapter 3 that the response value functions induced by PGRs can be informative evaluation techniques; however, it is reasonable that we might sometimes want to use response value functions which are not PGRs. One motivating example that we will explore more in the next chapter is Monte Carlo techniques. These response functions work by drawing some number of samples from the opponent's strategy, and using the results in some way to generate a counter strategy. They thus meet both of the criteria we were looking for in informative response functions: they in some way exploit the opponent (by basing the response on the opponent samples), and they do so with varying exploitative power (by varying the number of samples). Unfortunately, we can't guarantee that a Monte Carlo response will be a PGR since their behavior is stochastic.

4.2.1 Bounded Utility Deviations

To characterize how much a response function deviates from a pretty-good response, we introduce the concept of a **δ-pretty-good response**, which bounds how much utility the responder can lose by using the response function on the opponent's actual strategy instead of some other opponent strategy.

Definition 14. *A function $f\colon \Sigma_{-i} \to \Sigma_i$ is called a* **δ-pretty-good response** *if $u_i(\sigma_{-i}, f(\sigma_{-i})) + \delta \geq u_1(\sigma_{-i}, f(\sigma'_{-i}))$ for all $\sigma_{-i}, \sigma'_{-i} \in \Sigma_{-i}$.*

It should be noted that, because utilites are bounded, every response function is a δ-pretty-good response for some δ. Because we are dealing with functions that deviate from the PGR condition, we also need to handle deviations from no-regret learnability. To do so, we classify response functions by how close to an optimal response a regret minimization algorithm can get.

Definition 15. *Let $f\colon \Sigma_1 \to \Sigma_2$ be a response function and define $\sigma_2^t = f(\sigma_1^t)$ for $t = 1, ..., T$. f is called* **no-regret δ-learnable** *if for every sequence of strategies $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ such that $R_1^T \leq \varepsilon$ (where $\varepsilon > 0$), we have that*

$$u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + \varepsilon + \delta \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, f(\sigma_1^*)).$$

We show that the bound on deviation from the PGR condition is directly related to the bound on deviation from no-regret learnability.

Theorem 8. *Every $f\colon \Sigma_1 \to \Sigma_2$ that is a δ-pretty-good response is no-regret 2δ-learnable.*

Proof.

$$\max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) = \frac{1}{T} \max_{\sigma_1^* \in \Sigma_1} \sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^*)) \tag{4.13}$$

$$\leq \frac{1}{T} \max_{\sigma_1^* \in \Sigma_1} \sum_{t=1}^{T} \big(u_1(\sigma_1^*, f(\sigma_1^t)) + \delta\big) \tag{4.14}$$

$$\leq \frac{1}{T} \sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) + \varepsilon + \delta \tag{4.15}$$

$$\leq \frac{1}{T} \sum_{t=1}^{T} \big(u_1(\sigma_1^t, f(\overline{\sigma}_1^T)) + \delta\big) + \varepsilon + \delta \tag{4.16}$$

$$= u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + \varepsilon + 2\delta \tag{4.17}$$

□

This also applies to the high probability bound for the current strategies.

Theorem 9. *Let $f\colon \Sigma_1 \to \Sigma_2$ be a δ-pretty-good response and $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ be a sequence of strategies such that $R_1^T \leq \varepsilon$ when $\sigma_2^t = f(\sigma_1^t)$ for $t = 1, ..., T$. Then if we select T^* uniformly at random from $[1, T]$, we get that*

$$u_1(\overline{\sigma}_1^{T^*}, f(\overline{\sigma}_1^{T^*})) + \frac{\varepsilon + \delta}{p} \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, f(\sigma_1^*))$$

with probability at least $1 - p$ for any $p \in (0, 1]$.

Proof. We use (4.15) from the proof of Theorem 8 instead of (4.6) in a modification of the proof to Theorem 7. We also change the definition of k to be the number of indices such that $u_1(\sigma_1^t, f(\sigma_1^t)) < \max_{\sigma_1^* \in \Sigma} u_1(\sigma_1^*, f(\sigma_1^*)) - \frac{\varepsilon+\delta}{p}$. □

The current strategy bound is only proportional to δ, and thus is tighter than the 2δ in the average strategy bound. This is because the average strategy bound has to account for a potential deviation between $\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1, f(\sigma_1^t))$ and $u_1(\sigma_1, f(\overline{\sigma}_1^T))$, whereas the current strategy bound doesn't have to apply f to an averaged strategy. If we know that f is a linear map, then $\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1, f(\sigma_1^t)) = u_1(\sigma_1, f(\overline{\sigma}_1^T))$ follows by the linearity of expectation and the 2δ term in Theorem 8 can be replaced by δ.

4.2.2 Stochastic Response Functions

Although every response function is captured under the δ-pretty good response theory for some δ, the δ-PGR condition might not always be a good measure of how far the function deviates from PGR behavior. For example, a stochastic response function might violate the PGR condition by a large difference in utility, but only with a small probability. We now consider functions that meet the PGR condition in expectation or with high probability.

A **stochastic response function** maps a strategy σ_1 to a probability distribution over Σ_2. Equivalently, we can work with a probability distribution over response functions $F \in \Delta_{\{f\colon \Sigma_1 \to \Sigma_2\}}$. The idea of a response value function still holds for such distributions: $v_F(\sigma_1) = E_{f\sim F}[u_1(\sigma_1, f(\sigma_1))]$. The CFR-$f$ algorithm is extended to sample a response function $f_t \sim F$ on each iteration. We must also extend our notions of pretty-good response and no-regret learnable.

Definition 16. *Let $F \in \Delta_{\{f\colon \Sigma_{-i} \to \Sigma_i\}}$ be a probability distribution over response functions. We say that F is an* **expected pretty-good response** *if*

$$E_{f\sim F}[u_i(\sigma_{-i}, f(\sigma_{-i}))] \geq E_{f\sim F}[u_i(\sigma_{-i}, f(\sigma'_{-i}))] \tag{4.18}$$

for all $\sigma_{-i}, \sigma'_{-i} \in \Sigma_{-i}$. We say that F is an **expected $\boldsymbol{\delta}$-pretty-good response** *if*

$$E_{f\sim F}\left[u_i(\sigma_{-i}, f(\sigma_{-i}))\right] + \delta \geq E_{f\sim F}\left[u_i(\sigma_{-i}, f(\sigma'_{-i}))\right] \tag{4.19}$$

for all $\sigma_{-i}, \sigma'_{-i} \in \Sigma_{-i}$.

Definition 17. *Let $F \in \Delta_{\{f\colon \Sigma_1 \to \Sigma_2\}}$ be a probability distribution over response functions, and define $\sigma_2^t = f_t(\sigma_1^t)$ for $t = 1, ..., T$, where each $f_t \sim F$ is chosen independently. F is called* **no-regret learnable** *if for every sequence of strategies $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ such that $R_1^T \leq \varepsilon$, we have that*

$$E_{f\sim F}\left[u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T))\right] + \varepsilon \geq \max_{\sigma_1^* \in \Sigma_1} E_{f\sim F}\left[u_1(\sigma_1^*, f(\sigma_1^*))\right]$$

F is called **no-regret $\boldsymbol{\delta}$-learnable** *if we choose $f_1, ..., f_T \sim F$ independently and for every sequence of strategies $\sigma_1^1, \sigma_1^2, ..., \sigma_1^T \in \Sigma_1$ such that $R_1^T \leq \varepsilon$, we*

have that

$$E_{f\sim F}\left[u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T))\right] + \varepsilon + \delta \geq \max_{\sigma_1^* \in \Sigma_1} E_{f\sim F}\left[u_1(\sigma_1^*, f(\sigma_1^*))\right]$$

Let $u_{\max} = \max_{\sigma_1\in\Sigma_1,\sigma_2\in\Sigma_2} u_1(\sigma_1,\sigma_2)$ be the maximum expected utility that player 1 can achieve and $u_{\min} = \min_{\sigma_1\in\Sigma_1,\sigma_2\in\Sigma_2} u_1(\sigma_1,\sigma_2)$ be the minimum expected utility that player 1 can achieve (so $\Delta_1 = u_{\max} - u_{\min}$). We now show that expected pretty-good responses are no-regret learnable.

Theorem 10. *Every F that is an expected δ-pretty-good response is no-regret $(2\delta+\gamma)$-learnable with probability at least $1 - 2\exp(-\frac{T^2\gamma^2}{2\Delta_1^2})$, for any $\gamma > 0$.*

Proof. Let $\sigma_1^1, ..., \sigma_1^T$ be a sequence of strategies such that $R_1^T \leq \varepsilon$, and let $f_t \sim F$ be sampled independently for $t = 1, ..., T$.

Let $\sigma_1^* \in \operatorname{argmax}_{\sigma_1\in\Sigma_1} E_{f\sim F}[u_1(\sigma_1, f(\sigma_1))]$. If we assume each of the following:

$$E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^t))\right] \leq \frac{1}{T}\left[\sum_{t=1}^{T} u_1(\sigma_1^*, f_t(\sigma_1^t))\right] + \frac{1}{2}\gamma \tag{4.20}$$

$$\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f_t(\sigma_1^t)) \leq E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t))\right] + \frac{1}{2}\gamma \tag{4.21}$$

then it follows that F is no-regret $(2\delta+\gamma)$-learnable:

$$\begin{aligned}
\max_{\sigma_1^*\in\Sigma_1} E_{f\sim F}\left[u_1(\sigma_1^*, f(\sigma_1^*))\right] &= \frac{1}{T}\sum_{t=1}^{T} E_{f\sim F}\left[u_1(\sigma_1^*, f(\sigma_1^*))\right] && (4.22)\\
&\leq \frac{1}{T}\left[\sum_{t=1}^{T} E_{f\sim F}\left[u_1(\sigma_1^*, f(\sigma_1^t))\right]\right] + \delta && (4.23)\\
&= E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^t))\right] + \delta && (4.24)\\
&\leq \frac{1}{T}\left[\sum_{t=1}^{T} u_1(\sigma_1^*, f_t(\sigma_1^t))\right] + \delta + \frac{1}{2}\gamma && (4.25)\\
&\leq \frac{1}{T}\left[\sum_{t=1}^{T} u_1(\sigma_1^t, f_t(\sigma_1^t))\right] + \varepsilon + \delta + \frac{1}{2}\gamma && (4.26)\\
&\leq E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t))\right] + \varepsilon + \delta + \gamma && (4.27)\\
&\leq E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\overline{\sigma}_1^T))\right] + \varepsilon + 2\delta + \gamma && (4.28)\\
&= E_{f\sim F}\left[u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T))\right] + \varepsilon + 2\delta + \gamma && (4.29)
\end{aligned}$$

Thus we can bound the overall probability that F is not no-regret $(2\delta + \gamma)$-learnable by the probability that either (4.20) or (4.21) is false. Because $f_1, ..., f_T$ are chosen independently, we can do this using Hoeffding's Inequality.

$$\Pr\left[E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^*, f(\sigma_1^t))\right] - \frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^*, f_t(\sigma_1^t)) > \frac{1}{2}\gamma\right] \leq \exp\left(-\frac{T^2\gamma^2}{2\Delta_1^2}\right) \tag{4.30}$$

$$\Pr\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f_t(\sigma_1^t)) - E_{f\sim F}\left[\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t))\right] \geq \frac{1}{2}\gamma\right] \leq \exp\left(-\frac{T^2\gamma^2}{2\Delta_1^2}\right) \tag{4.31}$$

The probability that either event is true is no more than the sum of their individual probabilities, which gives us the result. □

Even if F is not a pretty-good response in expectation, it can be no-regret learnable if there is only a low probability p that a sample from it is not a pretty-good response. In this case, however, CFR-f does not converge fully to the optimal strategy according to v_F, but is only guaranteed to converge to a strategy that is within $2p\Delta_1$ of optimal.

Theorem 11. *Let $F \in \Delta_{\{f\colon \Sigma_1 \to \Sigma_2\}}$ be a probability distribution over response functions such that for any $\sigma_1, \sigma_1' \in \Sigma_1$, we have that $u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))$ with probability at least $1 - p$ given that $f \sim F$. Then F is no-regret $(2p\Delta_1 + \gamma)$-learnable with probability at least $1 - 2\exp(-\frac{T^2\gamma^2}{2\Delta_1^2})$, where γ is a free parameter.*

Proof.

$$
\begin{aligned}
&E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1))\right] \\
&\qquad \leq E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))|u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right] \\
&\qquad\qquad * \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right] \\
&\qquad\quad + u_{\max}\left(1 - \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right]\right) && (4.32) \\
&\qquad = E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))\right] \\
&\qquad\quad - E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))|u_1(\sigma_1, f(\sigma_1)) > u_1(\sigma_1, f(\sigma_1'))\right] \\
&\qquad\qquad * \left(1 - \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right]\right) \\
&\qquad\quad + u_{\max}\left(1 - \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right]\right) && (4.33) \\
&\qquad \leq E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))\right] \\
&\qquad\quad - u_{\min}\left(1 - \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right]\right) \\
&\qquad\quad + u_{\max}\left(1 - \Pr\left[u_1(\sigma_1, f(\sigma_1)) \leq u_1(\sigma_1, f(\sigma_1'))\right]\right) \\
&\qquad \leq E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))\right] + \Delta_1\left(1 - (1-p)\right) && (4.34) \\
&\qquad = E_{f\sim F}\left[u_1(\sigma_1, f(\sigma_1'))\right] + p\Delta_1 && (4.35)
\end{aligned}
$$

Thus F is an expected $p\Delta_1$-pretty-good response, and the result follows by Theorem 10. □

4.3 A Generalization of Restricted Nash Responses

As we discussed in Section 2.6, Restricted-Nash Responses are a technique for creating strategies that do well against an opponent model while limiting the RNR strategy's own exploitability. Theorem 5 shows what happens when we have a best response to an opponent which mixes between a static strategy and a best-response. We generalize this theorem to allow for any response functions.

For two response function $f, g\colon \Sigma_1 \to \Sigma_2$, define $f \circ^p g$ to be their mixture such that $(f \circ^p g)(\sigma_1) = f(\sigma_1)$ with probability p and $(f \circ^p g)(\sigma_1) = g(\sigma_1)$ with probability $1-p$.[2] Define

$$\Sigma_1^{f,\varepsilon-\text{safe}} = \{\sigma_1 \in \Sigma_1 \colon u_1(\sigma_1, f(\sigma_1)) \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, f(\sigma_1^*)) - \varepsilon\} \tag{4.36}$$

to be the set of strategies that achieve a utility within ε of optimal when played against an opponent that responds with f. For a best-response function BR, $\Sigma_1^{BR,\varepsilon-\text{safe}}$ is the same as $\Sigma_1^{\varepsilon-\text{safe}}$, the set of ε-safe strategies as presented in [23].

Theorem 12. *For any response functions f and g and any $p \in (0,1]$, if there is an $\phi > 0$ and a $\sigma_1 \in \Sigma_1$ such that*

$$u_1(\sigma_1, (f \circ^p g)(\sigma_1)) \geq u_1(\sigma_1', (f \circ^p g)(\sigma_1')) - \phi \tag{4.37}$$

for all $\sigma_1' \in \Sigma_1$, then there is some ε such that $\sigma_1 \in \Sigma_1^{g,\varepsilon-safe}$ and

$$u_1(\sigma_1, f(\sigma_1)) \geq \max_{\sigma_1^* \in \Sigma_1^{g,\varepsilon-safe}} u_1(\sigma_1^*, f(\sigma_1^*)) - \phi \tag{4.38}$$

Proof. Let $\varepsilon = \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, g(\sigma_1^*)) - u_1(\sigma_1, g(\sigma_1))$. Clearly for this ε, we have $\sigma_1 \in \Sigma_1^{g,\varepsilon-\text{safe}}$. Then we get:

$$\begin{aligned} u_1(\sigma_1, (f \circ^p g)(\sigma_1)) &= p u_1(\sigma_1, f(\sigma_1)) + (1-p) u_1(\sigma_1, g(\sigma_1)) && (4.39)\\ &= p u_1(\sigma_1, f(\sigma_1)) + (1-p)(\max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, g(\sigma_1^*)) - \varepsilon) && (4.40)\end{aligned}$$

Pluggin any arbitrary $\sigma_1' \in \Sigma_1^{g,\varepsilon-\text{safe}}$ into (4.37), we also get:

$$\begin{aligned} u_1(\sigma_1, (f \circ^p g)(\sigma_1)) &\geq u_1(\sigma_1', (f \circ^p g)(\sigma_1')) - \phi && (4.41)\\ &= p u_1(\sigma_1', f(\sigma_1')) + (1-p) u_1(\sigma_1', g(\sigma_1')) - \phi && (4.42)\\ &\geq p u_1(\sigma_1', f(\sigma_1')) + (1-p)(u_1(\sigma_1^*, g(\sigma_1^*)) - \varepsilon) - \phi && (4.43)\end{aligned}$$

By subtracting (4.40) from (4.43) and rearranging, we get $u_1(\sigma_1, f(\sigma_1)) \geq u_1(\sigma_1', f(\sigma_1')) - \phi$. Because this is true for every $\sigma_1' \in \Sigma_1^{g,\varepsilon-\text{safe}}$, we get our desired result. □

[2] As we saw when discussing an average strategy (Section 2.1.1), we can also find a behavioral strategy equivalent to $(f \circ^p g)(\sigma_1)$ by performing a weighted average of $f(\sigma_1)$ and $g(\sigma_1)$ at each information set.

By choosing f to output a static strategy ($f(\sigma_1) = \sigma_2$) and g to be a best-response ($g(\sigma_1) \in \arg\max_{\sigma_2^* \in \Sigma_2} u_2(\sigma_1, \sigma_2^*)$), we see that Theorem 5 is a special case of Theorem 12. Theorem 12 thus allows us to generalize a tradeoff between exploitation and exploitability to a tradeoff between performance by any two response value functions. In order to find the strategies that allow such a tradeoff, we must find strategies that satisfy the assumption in Theorem 12. We can do this with a modified form of the RNR algorithm.

Corollary 1. *Let $\sigma_1^1, ..., \sigma_1^T$ be a sequence of strategies for player 1 such that $R_1^T \leq \phi$ (where phi > 0). For any pretty-good response functions f and g, let the strategies for player 2 be defined such that $\sigma_2^t = (f \circ^p g)(\sigma_1^t)$ for each time step $t = 1, ..., T$. Then there is some ε such that $\overline{\sigma}_1^T \in \Sigma_1^{g,\varepsilon-safe}$ and*

$$u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) \geq \max_{\sigma_1^* \in \Sigma_1^{g,\varepsilon-safe}} u_1(\sigma_1^*, f(\sigma_1^*)) - \phi \tag{4.44}$$

Proof. Because f and g are pretty-good responses, so is $f \circ^p g$:

$$\begin{align} u_1(\sigma_1, (f \circ^p g)(\sigma_1) &= pu_1(\sigma_1, f(\sigma_1)) + (1-p)u_1(\sigma_1, g(\sigma_1)) \tag{4.45} \\ &\leq pu_1(\sigma_1, f(\sigma_1')) + (1-p)u_1(\sigma_1, g(\sigma_1')) \tag{4.46} \\ &= u_1(\sigma_1, (f \circ^p g)(\sigma_1')) \tag{4.47} \end{align}$$

Then by Theorem 6, we know that

$$u_1(\overline{\sigma}_1^T, (f \circ^p g)(\overline{\sigma}_1^T)) \geq \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, (f \circ^p g)(\sigma_1^*)) - \phi \tag{4.48}$$

and the result follows by Theorem 12. □

This corollary shows that if we run CFR-($f \circ^p g$) for pretty-good responses f and g, the CFR player's average strategy converges to a strategy that maximizes performance against f within a set of strategies with bounded loss against g. Again, by using a static strategy for f and a best-response for g, we get an algorithm which produces RNR strategies.

In Chapter 3 we showed that PGRs can be useful as an evaluation tool, either replacing or supplementing exploitability. This corollary also shows how we can use the response value function v_f induced by a PGR f as a metric in the RNR technique instead of exploitability. By having player 2 mix between a static strategy σ_2 and a pretty-good response $f(\sigma_1^t)$, we see that $\overline{\sigma}_1^T$ converges to a strategy that is a best-response against σ_2 within the set $\Sigma_1^{f,\varepsilon-\text{safe}}$. Thus we can find a strategy that exploits σ_2 while bounding its loss against any specific PGR, not just a best-response.

4.3.1 Regret Minimization and Best-Responses

Above, we showed convergence if player 2 mixes between a static strategy and a best-response. In the actual RNR implementation, though, the RNR opponent uses a regret minimization algorithm (i.e. CFR) instead of a best-response, because CFR iterations are computationally cheaper and the algorithm converges faster. Noting again that an f that always returns a static strategy ($f(\sigma_1) = \sigma_2$) is a pretty-good response, we generalize the RNR algorithm by having the response player mix between a regret minimization algorithm and a pretty-good response. Ultimately, we show that this algorithm converges as if the opponent had used a best-response whenever he actually used the regret minimization algorithm. From the results in the previous section, this thus lets us find ε-safe strategies that maximize performance against PGRs, while avoiding the cost of running a best-response computation on each iteration.

Because we are now dealing with a regret minimization algorithm that is only run probabilistically on each iteration, we introduce a new notation instead of discussing sequences that minimize regret. During an iterated game over T time steps, a regret minimization algorithm $\mathcal{A}$ operates by specifying a strategy $\sigma_{i,\mathcal{A}}^t$ for player i at each time $t = 1, ..., T$. The choice of $\sigma_{i,\mathcal{A}}^t$ is a function of each of the observed previous utilities $u_i(\sigma_{i,\mathcal{A}}^{t'}, \sigma_{-i}^{t'})$ for $t' = 1, ..., t-1$. In the original RNR implementation, the opponent only runs $\mathcal{A}$ on each iteration with probability $1-p$. Formally, we can say the RNR opponent samples a set of time steps $\mathcal{T} \subseteq \{1, ..., T\}$ such that $t \in \mathcal{T}$ with independent probability $1 - p$ for each $t \in \{1, ..., T\}$. We say $K = |\mathcal{T}|$ is the number of such sampled time steps (so $\mathbb{E}[K] = (1-p)T$), and label the elements of $\mathcal{T}$ as $t_{j_1}, ..., t_{j_K}$. Then for the RNR opponent, $\sigma_2^{t_{j_k}} = \sigma_{2,\mathcal{A}}^{j_k}$ for each $t_{j_1} \in \mathcal{T}$, and $\mathcal{A}$ only receives the inputs $u_2(\sigma_1^{t_{j_{k'}}}, \sigma_{2,\mathcal{A}}^{j_{k'}})$ for $k' = 1, ..., k-1$. On time steps

$t \notin \mathcal{T}$, the opponent sets $\sigma_2^t = \sigma_2'$, where σ_2' is the fixed opponent model played by the opponent with probability p, and $\mathcal{A}$ receives no input.

Using this notation, we prove the following lemma which is useful when analyzing players that do regret minimization on each iteration with some probability less than 1.

Lemma 2. *Let $\mathcal{A}$ be a regret algorithm that guarantees $R_2^K \leq \varepsilon$ (where $\varepsilon > 0$). Let σ_2^t be chosen on each time step $t = 1, ..., T$ such that with probability $0 < p \leq 1$, σ_2^t is assigned by regret minimization algorithm $\mathcal{A}$, and with probability $1 - p$, σ_2^t is set to any arbitrary strategy. Then the following holds*

$$\frac{1}{T} \max_{\sigma_2^* \in \Sigma_2} \sum_{t=1}^{T} \left(u_2(\sigma_1^t, \sigma_2^*) - u_2(\sigma_1^t, \sigma_{2,\mathcal{A}}^t) \right) \leq \varepsilon \tag{4.49}$$

where $\sigma_{2,\mathcal{A}}^t$ is the strategy that $\mathcal{A}$ would select on iteration t if $\mathcal{A}$ were queried.

Proof. Because $\mathcal{T}$ is a random sample of $\{1, ..., T\}$ we have that for any $\sigma_2'' \in \Sigma_2$

$$\begin{aligned}
\frac{1}{K} \max_{\sigma_2^* \in \Sigma_2} \sum_{k=1}^{K} u_2(\sigma_1^{t_{j_k}}, \sigma_2^*) &\geq \frac{1}{K} \sum_{k=1}^{K} u_2(\sigma_1^{t_{j_k}}, \sigma_2'') && (4.50)\\
&= \mathbb{E}_{\mathcal{T}} \left[\frac{1}{K} \sum_{t \in \mathcal{T}} u_2(\sigma_1^t, \sigma_2'') \right] && (4.51)\\
&= \frac{1}{T} \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2'') && (4.52)
\end{aligned}$$

Step 4.51 is due to the linearity of expectation, step 4.52 is because the expected mean of a sample is equal to the mean of the set sampled from. Because this is true for every $\sigma_2'' \in \Sigma_2$, it must be the case that

$$\frac{1}{K} \max_{\sigma_2^* \in \Sigma_2} \sum_{k=1}^{K} u_2(\sigma_1^{t_{j_k}}, \sigma_2^*) \geq \frac{1}{T} \max_{\sigma_2^* \in \Sigma_2} \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^*) \tag{4.53}$$

Again, because the expected mean of a sample is equal to the mean of the set sampled from, we have

$$\frac{1}{K} \sum_{k=1}^{K} u_2(\sigma_1^{t_{j_k}}, \sigma_{2,\mathcal{A}}^{t_{j_k}}) = \frac{1}{T} \sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_{2,\mathcal{A}}^t) \tag{4.54}$$

The regret minimization guarantee of $\mathcal{A}$ shows

$$\frac{1}{K}\max_{\sigma_2^*\in\Sigma_2}\sum_{k=1}^{K}\Big(u_2(\sigma_1^{t_{j_k}},\sigma_2^*)-u_2(\sigma_1^{t_{j_k}},\sigma_{2,\mathcal{A}}^{t_{j_k}})\Big)\leq\varepsilon \tag{4.55}$$

By applying (4.53) to the left side of the sum and (4.54) to the right side of the sum, we get the result. □

Using the lemma, we can now analyze an algorithm that mixes between regret minimization and a pretty-good response.

Theorem 13. *Let $\mathcal{A}$ be a regret algorithm that guarantees $R_i^T \leq \varepsilon_i$ (where $\varepsilon_i > 0$ and $\varepsilon_i = o(1)$), and let $f\colon \Sigma_1 \to \Sigma_2$ be a pretty-good response. For $t = 1, ..., T$ let $\sigma_1^t = \sigma_{1,\mathcal{A}}^t$ be assigned by $\mathcal{A}$, and let σ_2^t be chosen such that $\sigma_2^t = f(\sigma_1^t)$ with probability $p \in (0,1]$ and $\sigma_2^t = \sigma_{2,\mathcal{A}}^t$ with probability $1-p$. Then the average strategy for player 1 satisfies*

$$u_1(\overline{\sigma}_1^T,(f\circ^p BR)(\overline{\sigma}_1^T))\geq u_1(\sigma_1',(f\circ^p BR)(\sigma_1'))-|o(1)| \tag{4.56}$$

for all $\sigma_1' \in \Sigma_1$, where BR is a best-response function defined as $BR(\sigma_1) = \arg\max_{\sigma_2^*\in\Sigma_2} u_2(\sigma_1,\sigma_2^*)$.

Proof. We start by proving the following:

$$u_1(\overline{\sigma}_1^T,\overline{\sigma}_2^T)\geq\max_{\sigma_1^*\in\Sigma_1}u_1(\sigma_1^*,\overline{\sigma}_2^T)-|o(1)| \tag{4.57}$$

$$u_2(\overline{\sigma}_1^T,\overline{\sigma}_2^T)\geq pu_2(\overline{\sigma}_1^T,f(\overline{\sigma}_1^T))+(1-p)(\max_{\sigma_2^*\in\Sigma_2}u_2(\overline{\sigma}_1^T,\sigma_2^*))-|o(1)| \tag{4.58}$$

These equations show that $\overline{\sigma}_1^T$ converges to a best-response to $\overline{\sigma}_2^T$, and $\overline{\sigma}_2^T$ converges to a mixture between $f(\overline{\sigma}_1^T)$ and a best-response.

We first prove (4.57).

$$\begin{aligned}
u_1(\overline{\sigma}_1^T, \overline{\sigma}_2^T) &= \frac{1}{T}\sum_{t=1}^{T} u_1(\overline{\sigma}_1^T, \sigma_2^t) && (4.59)\\
&= \frac{1}{T}\sum_{t=1}^{T} \big(pu_1(\overline{\sigma}_1^T, f(\sigma_1^t)) + (1-p)u_1(\overline{\sigma}_1^T, \sigma_{2,\mathcal{A}}^t)\big) && (4.60)\\
&\geq p\frac{1}{T}\sum_{t=1}^{T} u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + (1-p)\frac{1}{T}\min_{\sigma_2^* \in \Sigma^2}\sum_{t=1}^{T} u_1(\overline{\sigma}_1^T, \sigma_2^*) && (4.61)\\
&\geq pu_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + (1-p)\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_{2,\mathcal{A}}^t) - (1-p)|o(1)| && (4.62)\\
&= p\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\overline{\sigma}_1^T)) + (1-p)\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_{2,\mathcal{A}}^t) - |o(1)| && (4.63)\\
&\geq p\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, f(\sigma_1^t)) + (1-p)\frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_{2,\mathcal{A}}^t) - |o(1)| && (4.64)\\
&= \frac{1}{T}\sum_{t=1}^{T} u_1(\sigma_1^t, \sigma_2^t) - |o(1)| && (4.65)\\
&\geq \frac{1}{T}\max_{\sigma_1^* \in \Sigma_1}\sum_{t=1}^{T} u_1(\sigma_1^*, \sigma_2^t) - |o(1)| - |o(1)| && (4.66)\\
&= \max_{\sigma_1^* \in \Sigma_1} u_1(\sigma_1^*, \overline{\sigma}_2^T) - |o(1)| && (4.67)
\end{aligned}$$

Steps 4.61 and 4.64 use the PGR condition on the left sum. Step 4.62 uses Lemma 2 and that $\varepsilon_2 = o(1)$, and step 4.66 uses that $R_1^T \leq \varepsilon_1$ and $\varepsilon_1 = o(1)$.

We now prove (4.58)

$$\begin{aligned}
u_2(\overline{\sigma}_1^T, \overline{\sigma}_2^T) &= \frac{1}{T}\sum_{t=1}^{T} u_2(\overline{\sigma}_1^T, \sigma_2^t) && (4.68)\\
&\geq \frac{1}{T}\min_{\sigma_1^*\in\Sigma_1}\sum_{t=1}^{T} u_2(\sigma_1^*, \sigma_2^t) && (4.69)\\
&\geq \frac{1}{T}\sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^t) - |o(1)| && (4.70)\\
&= \frac{1}{T}\sum_{t=1}^{T} \big(pu_2(\sigma_1^t, f(\sigma_1^t)) + (1-p)u_2(\sigma_1^t, \sigma_{2,\mathcal{A}}^t)\big) - |o(1)| && (4.71)\\
&\geq p\frac{1}{T}\sum_{t=1}^{T} u_2(\sigma_1^t, f(\overline{\sigma}_1^T))\\
&\qquad + (1-p)\frac{1}{T}\max_{\sigma_2^*\in\Sigma_2}\sum_{t=1}^{T} u_2(\sigma_1^t, \sigma_2^*) - (1-p)|o(1)| - |o(1)|\\
& && (4.72)\\
&= pu_2(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + (1-p)(\max_{\sigma_2^*\in\Sigma_2} u_1(\overline{\sigma}_1^T, \sigma_2^*)) - |o(1)| && (4.73)
\end{aligned}$$

Step 4.70 uses $R_1^T \leq \varepsilon_1$ and $\varepsilon_1 = o(1)$, and step 4.72 uses the PGR condition on the left sum and Lemma 2 on the right sum.

Using (4.57) and (4.58) we now show the following for arbitrary $\sigma_1' \in \Sigma_1$:

$$\begin{aligned}
&pu_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) + (1-p)(\min_{\sigma_2^*\in\Sigma_2} u_1(\overline{\sigma}_1^T, \sigma_2^*))\\
&\qquad\geq u_1(\overline{\sigma}_1^T, \overline{\sigma}_2^T) - |o(1)| && (4.74)\\
&\qquad\geq \max_{\sigma_1^*\in\Sigma_1} u_1(\sigma_1^*, \overline{\sigma}_2^T) - |o(1)| - |o(1)| && (4.75)\\
&\qquad\geq u_1(\sigma_1', \overline{\sigma}_2^T) - |o(1)| && (4.76)\\
&\qquad= \frac{1}{T}\sum_{t=1}^{T} \big(pu_1(\sigma_1', f(\sigma_1^t) + (1-p)u_1(\sigma_1', \sigma_{2,\mathcal{A}}^t)\big) - |o(1)| && (4.77)\\
&\qquad\geq \frac{1}{T}\sum_{t=1}^{T} \left(pu_1(\sigma_1', f(\sigma_1')) + (1-p)\min_{\sigma_2^*\in\Sigma_2} u_1(\sigma_1', \sigma_2^*)\right) && (4.78)\\
&\qquad= pu_1(\sigma_1', f(\sigma_1')) + (1-p)\min_{\sigma_2^*\in\Sigma_2} u_1(\sigma_1', \sigma_2^*) && (4.79)
\end{aligned}$$

The result follows by linearity of expectation. □

This shows that the average strategy for player 1 converges to a strategy that maximizes performance against an opponent that mixes between the pretty-good response f and a best-response.

Corollary 2. *Let $\mathcal{A}$ be a regret algorithm that guarantees $R_i^T \leq \phi$ (where $\phi > 0$ and $\phi = o(1)$), and let $f \colon \Sigma_1 \to \Sigma_2$ be a pretty-good response. For $t = 1, ..., T$ let $\sigma_1^t = \sigma_{1,\mathcal{A}}^t$ be assigned by $\mathcal{A}$, and let σ_2^t be chosen such that $\sigma_2^t = f(\sigma_1^t)$ with probability $p \in (0,1]$ and $\sigma_2^t = \sigma_{2,\mathcal{A}}^t$ with probability $1-p$. Then there is some ε such that $\sigma_1 \in \Sigma_1^{f,\varepsilon-safe}$ and*

$$u_1(\overline{\sigma}_1^T, f(\overline{\sigma}_1^T)) \geq \max_{\sigma_1^* \in \Sigma_1^{\varepsilon-safe}} u_1(\sigma_1^*, f(\sigma_1^*)) - |o(1)| \tag{4.80}$$

Proof. The proof follows immediately from applying Theorem 12 with the result from Theorem 13 □

From this corollary, we see that we can also generalize the RNR algorithm to be able to find strategies that maximize performance against a PGR while ensuring bounded exploitability.

Chapter 5

Empirical Results: Learning to Beat Opponents that Learn

In the previous chapter we showed that we can use CFR-f to learn static strategies that are close to optimal for playing against certain adaptive opponents. However, there are widely used response agents that do not fit neatly into the pretty-good response framework. These responses are δ-pretty-good responses for some δ, but it is hard to put a tight bound on δ. In this chapter, we empirically examine whether CFR-f can learn strong strategies against such response functions.

In particular, we want to examine response functions that are used in strong adaptive agents. The response function framework assumes that the response player has full access to the other player's strategy; without knowledge of σ_1, they can't compute $f(\sigma_1)$. However, typical play takes place in an online setting, where the only knowledge that player 2 can gain about σ_1 is by observing the actions it takes during play. Thus, any practical response function for online play must be able to take a set of samples as input, as opposed to a full strategy.

In order to approximate how an adaptive agent works with samples in the online setting, we examine Monte Carlo response functions. A Monte Carlo response function $f : \Sigma_1 \rightarrow \Sigma_2$ can be thought of as a composition of two functions $f_1 \colon \Sigma_1 \rightarrow \Psi$ and $f_2 \colon \Psi \rightarrow \Sigma_2$, where Ψ is the set of possible observations of strategies in Σ_1. f_1 samples a set of observations of σ_1, and then f_2 constructs a response strategy based on the observations. In the online setting, the responder would simply use f_2 based on the actual observations from previous time steps.

In addition to their similarity to adaptive online agents, we want to use Monte Carlo methods because they have properties which make them useful for strategy evaluation. In general, a Monte Carlo method will make some number of samples n of the opponent strategy σ_1. As the number of samples increases, the responder's knowledge of σ_1 increases, and thus he is better able to exploit his opponent. Thus, by choosing different values of n in a Monte Carlo method, we can actually generate a series of response functions which vary in their exploitative strength. This gives us a convenient way to measure multiple dimensions of strategy strength.

5.1 Monte Carlo Tree Search in Leduc Hold'em

Monte Carlo tree search (MCTS) is an online learning framework for extensive-form games. A MCTS algorithm builds an approximation of the game tree through repeated sampling, during which it uses two components. A **selection** rule is used to choose which action to make at player choice nodes where data is available from previous time steps, and a **rollout** player is used to make decisions once the game enters a state not encountered on previous time steps. Monte Carlo tree search methods only converge in single-player games or in games with perfect information (i.e., all information sets have size 1), and thus they can't be applied directly to general extensive-form games.

UCT is a MCTS algorithm that uses a selection rule adapted from the multi-armed bandit setting [28]. In particular, it selects the action with the largest upper confidence bound, as defined in the **UCB1** algorithm, which achieves optimal regret in the bandit setting where each action results in a utility sampled from a stationary distribution associated with the action [2]. As the number of training iterations goes to infinity, the strategy proposed by the UCT selection rule converges to optimal. In practice, UCT has been used to create strong agents in large extensive-form games, particularly Go [15].

5.1.1 Algorithm Design

By fixing a strategy σ_1 for player 1, we can turn an imperfect information two-player game into a perfect information one-player game. All of player 1's choice nodes are turned into chance nodes with probability distributions corresponding to σ_1. Any actions or chance events that are unobserved by player 2 can be thought of as happening only when player 2 observes the result (e.g. at a terminal node); Bayes' rule can be used to ensure all outcome possibilities are unchanged.

In this way, we turn UCT into a response function $\text{UCT}: \Sigma_1 \to \Sigma_2$. Given a static player strategy σ_1, UCT gets n iterations to explore the one-player game tree and learn the utilities of different actions. For the rollout policy, we use **always call** to rollout player 2's actions, as it has been shown to outperform other default policies [21, Figure 4.3]. At every information set, always call selects the call/check action. After the n training iterations, we select the strategy σ_2 which plays the action with the highest observed average utility during the training iterations. At information sets that the UCT training never reached, we again have σ_2 default to the call/check action with probability 1. Because the strategy output by UCT depends on the number of training iterations n, we denote the fully specified response function as UCT-n.

Now that we have a UCT response function, we can plug it into CFR-f, resulting in CFR-UCT-n. For each iteration t of CFR-UCT-n, we run a CFR update for player 1 create strategy σ_1^t, and then we apply response function UCT-n to generate $\sigma_2^t = \text{UCT-}n(\sigma_1^t)$. In order to ensure that UCT-n depends only on the current strategy σ_1^t, we wipe the UCT game tree on every CFR iteration, before running n UCT iterations from scratch.

5.1.2 Results

We trained CFR-UCT-n strategies in Leduc Hold'em using a variety of n values. To create a baseline, we also trained a strategy with CFR, resulting in an ε-Nash equilibrium exploitable for only 2 milli-bets per game (mb/g). Against each of these strategies, we ran the response function UCT-n with a variety of n values to create counter-strategies, and then measured how well the test strategies do when played against their respective counter-strategies. To reduce variance caused by the random sampling in UCT, we used 100 independent runs of UCT-n for each n value and averaged performance over the resulting strategies. 95% confidence intervals are not shown, but are on the order of ±1 mb/g for all values.

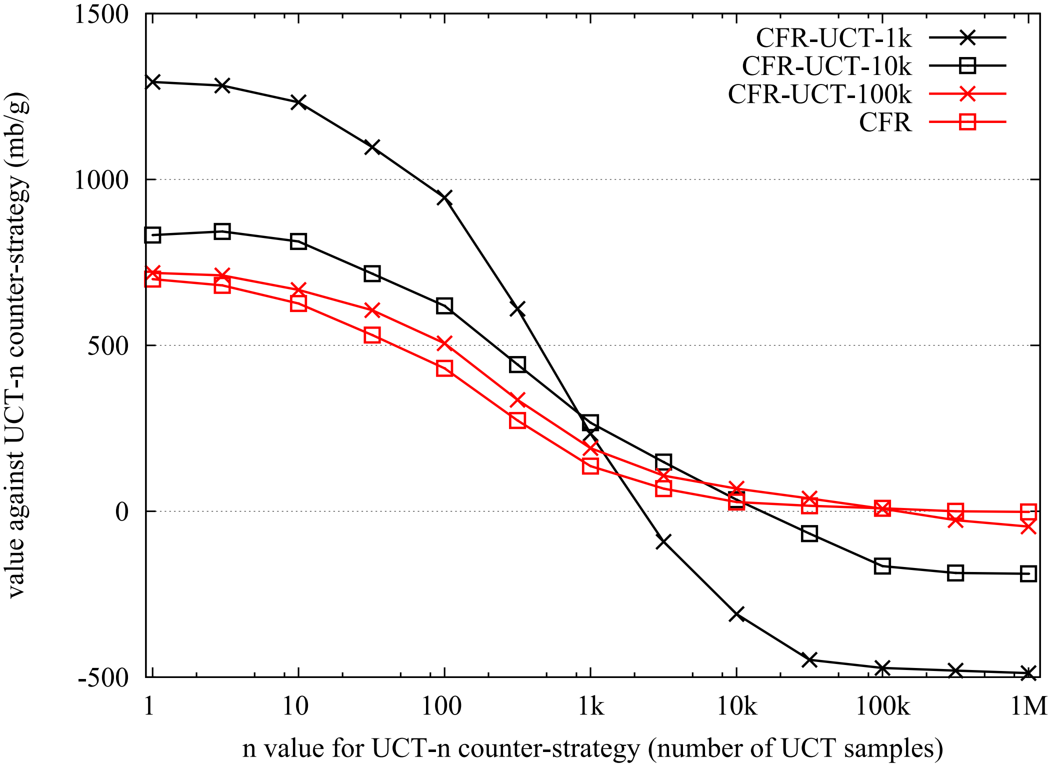

Figure 5.1: Performance of CFR-UCT-n strategies and a CFR strategy against UCT-n counter-strategies, as the counter-strategy n value is increased.

Figure 5.1 shows the results of playing CFR-UCT-1k, CFR-UCT-10k, CFR-UCT-100k, and the CFR strategy against UCT-n counter-strategies trained using a range of n values. Each line represents one of the strategies being tested, and the x-axis shows how many iterations UCT-n is given to learn a strategy for playing against each target strategy. Figure 5.2 shows much of the same data, but now with more n values for CFR-UCT-n represented on the x-axis, and also shows how the CFR-UCT strategies do in one-on-one play against the CFR strategy and against a best response. Each line represents a

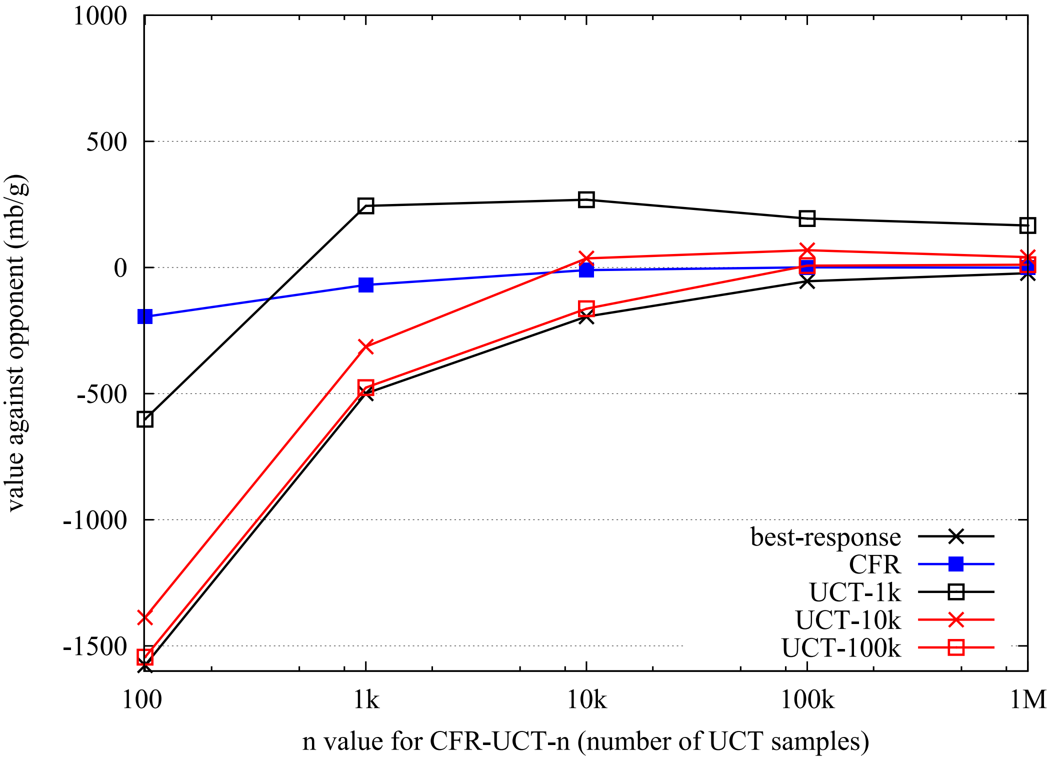

Figure 5.2: Performance of CFR-UCT-n strategies against a variety of opponents as the n value is increased.

particular opponent that we play our CFR-UCT-n strategies against.

From the results we can see that for any $n_2 \leq n_1$, the strategy produced by CFR-UCT-n_1 will achieve higher value than an ε-equilibrium in one-on-one play against UCT-n_2. We have thus shown that using CFR-f, we can learn to do better against adaptive opponents than we would do by playing an optimal strategy for the game. In addition, we have shown that CFR-f can converge to an effective strategy against an opponent not covered by the pretty-good response theory. Despite being highly exploitable, the CFR-UCT strategies lose only a small amount in one-on-one play against an ε-equilibrium (the CFR strategy), and do not lose to the weaker UCT counter-strategies which are actively trying to exploit them, indicating that they are difficult to exploit, a strength not shown via the exploitability metric. The UCT-n_2 response strategies are able to exploit the CFR-UCT-n_1 strategies only when n_2 is much larger than n_1, which lends credence to our conjecture that using a range of k values gives us a set of response functions which are able to exploit the opponent with varying strength.

5.2 Frequentist Best-Response in Texas Hold'em

The results using UCT in Leduc Hold'em were promising, and as a next step we wanted to investigate how CFR-f would scale to a large game such as Texas Hold'em. Preliminary tests showed that UCT doesn't scale well to Texas Hold'em. Even when we gave the UCT player over 50 minutes to learn a counter-strategy, resulting in over 50 million UCT iterations, the UCT player produced counter-strategies that still *lost* 41 mbb/g against a CFR strategy which was exploitable for 282 mbb/g. In order to run CFR-f in a timely manner, the f iterations would need to be faster than this, so UCT will be too weak of a response function to be considered a reasonable exploitative agent.

Instead, we use the technique of **Frequentist Best-Response (FBR)**, first introduced for use in poker games by Johanson et al. as a method for quickly approximating a best-response[23]. FBR works in two steps. First, it uses some form of Monte Carlo sampling to construct a model of the target strategy. Second, it computes a best-response to the model, thus approximating a best-response to the actual strategy. Because of sparsity and computation concerns, the model and best-response are generally constructed in some abstract game. If the target strategy is consistent with the FBR model abstraction, the model converges to the target strategy as the number of samples increases, and FBR thus converges to a (abstract game) best-response.

To compare how they scale, we ran FBR in a small abstraction to respond to the same CFR strategy that UCT couldn't beat after 50 million iterations. The FBR strategy needed less than 100,000 iterations and less than a minute of computation time to produce a strategy with positive expected utility against the CFR strategy. We conclude that (abstracted) FBR is a better method than (unabstracted) UCT for creating exploitative agents in Texas Hold'em.

5.2.1 Algorithm Design

Johanson explored different parameter choices for constructing a FBR [21]. Following his lead, we collect the sampled data for the model by playing simulated games between the target strategy and a **probe** strategy, which mixes between the call/check and bet/raise actions with equal probability. We also set the model's default strategy to always call; this will be the strategy played by the model in information sets that were never sampled from the target strategy. We label the response function that results from running FBR with n training iterations as FBR-n: $\Sigma_1 \to \Sigma_2$. When FBR-n is called on σ_1, it plays n hands of probe against σ_1, constructs a model from the resulting data, and then outputs a best-response to the model.

Using an FBR-n response function in CFR-f results in the algorithm CFR-FBR-n. Because a full tree walk is computationally infeasible in some abstractions, we make use of public chance sampling during the CFR iterations. This has the added benefit of making the CFR iterations faster, but it also means that the CFR iterations are much slower than the FBR-n iterations, which must make a full tree walk in the best-response abstraction. To balance the time spent on each player, we run 1200 PCS iterations parellelized over 24 CPUs to update the CFR strategy between each run of FBR-n.

5.2.2 Training with CFR-FBR

For computational reasons, we ran our CFR-FBR-n experiments in a small Texas Hold'em abstraction which groups hands into five percentile buckets on each betting round using expected hand strength squared. This abstraction contains 3.6×10^6 information sets, as compared to 1116 in Leduc Hold'em, so it is still useful for testing how CFR-f scales. In order to evaluate our techniques, we ran CFR in the same abstraction in order to produce an abstract game ε-Nash equilibrium, resulting in a strategy exploitable (in the abstract game) for only 4 mbb/g. Running CFR in the same abstraction which we are using to produce FBR makes sense not only because we can directly compare the resources used by CFR-UCT to the resources used by CFR, but also because we expect a CFR strategy specifically designed for our test abstraction to do well against responses in the same abstraction, thus giving us the strongest possible benchmark to compare against.

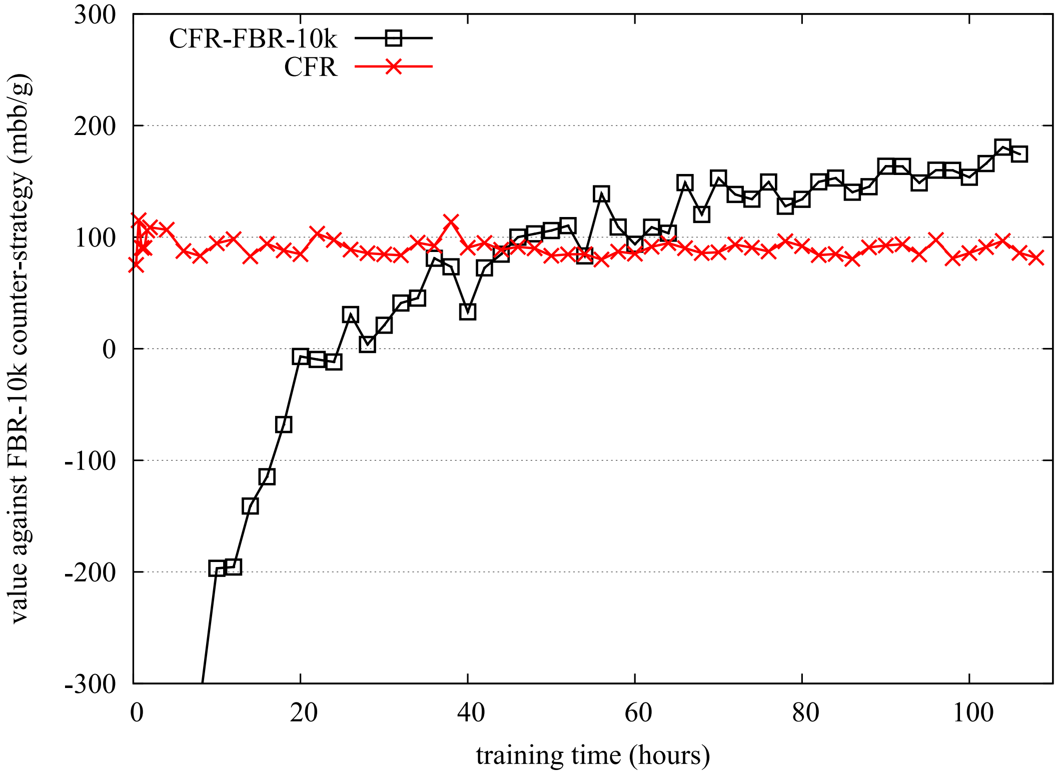

Figure 5.3: Performance of CFR-FBR-10k and CFR during training as measured by value against a FBR-10k adversary.

We began by training a strategy with CFR-FBR-10k. Over the course of running the algorithm, we had it periodically output the strategy it would return if stopped at that time, and we used FBR-10k to evaluate these checkpoint strategies. We similarly evaluated the CFR strategy at a number of checkpoints during CFR training. The results are shown in Figure 5.3.

From the graph we see that CFR very quickly produces a strategy that does well against the FBR-10k adversary, but then performance plateaus. In contrast, CFR-FBR-10k takes a comparatively long time to find a strategy that does not have very poor performance against the FBR-10k opponent; through 24 hours of computation, it is still exploitable for positive utility by this response function which loses 75 mbb/g to the CFR strategy after only

20 minutes of training. Given enough computation time, though, the CFR-FBR-10k strategy eventually passes the equilibrium in performance.

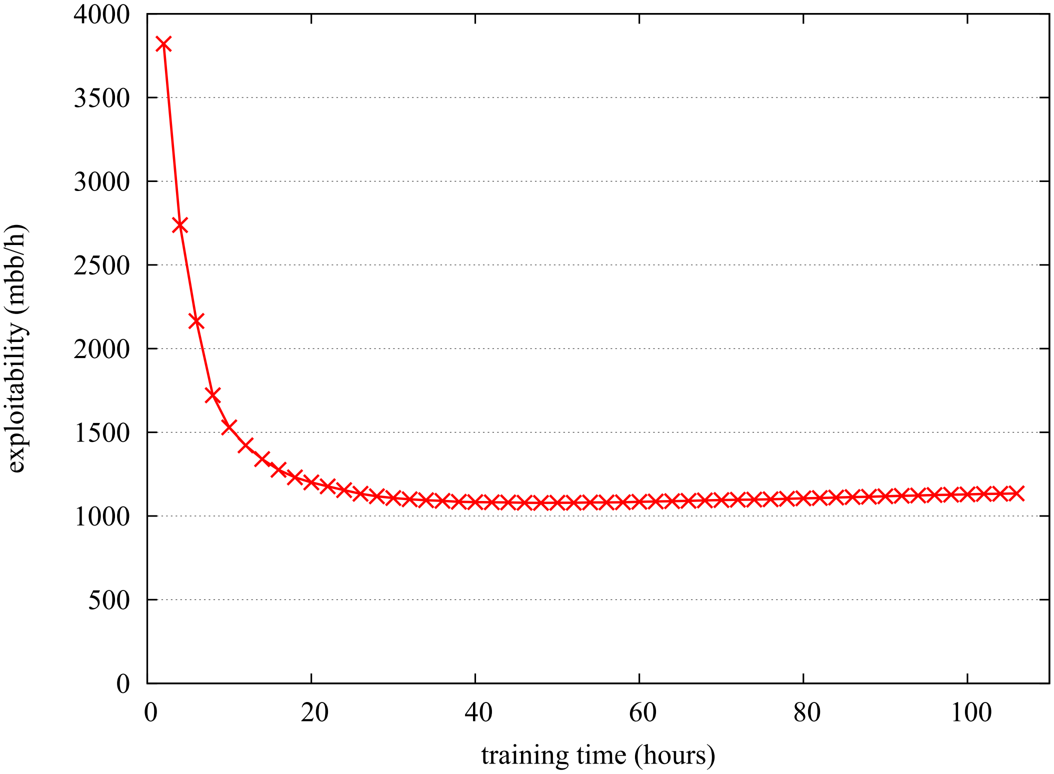

Figure 5.4: Abstract game exploitability of CFR-FBR-10k during training as the training time is increased.

Figure 5.4 shows the abstract game exploitability of the CFR-FBR-10k strategy over time. Although the CFR-FBR-10k strategy achieves performance comparable to CFR against FBR-10k, and eventually rates better by that evaluation metric, we can see from this graph that the CFR-FBR-10k strategy never even approaches the CFR strategy by the traditional metric of exploitability. The CFR-FBR-10k plateaus with an exploitability above 1000 mbb/g; in contrast, an **always fold** strategy, which simply surrenders its forced bet on every hand, is exploitable for 750 mbb/g. By comparing the two graphs, we can also see that the CFR-FBR-10k strategy continues to improve according to the FBR-10k metric even after its exploitability levels off and even slightly degrades. This clearly demonstrates that exploitability and the FBR metric are measuring different dimensions of strategy strength.

5.2.3 Results

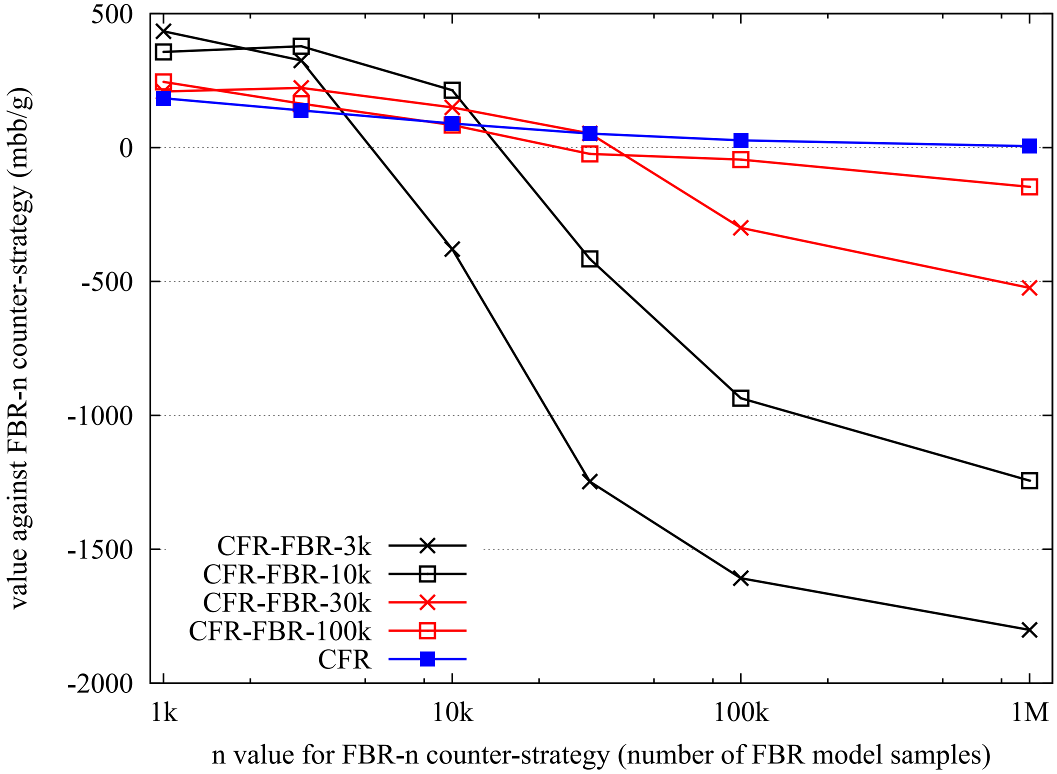

Figure 5.5: Performance of CFR-FBR-n strategies and a CFR strategy against FBR-n counter-strategies, as the counter-strategy n value is increased.

Similar to our evaluation of CFR-UCT, we ran CFR-FBR-n for a variety of n values to produce a set of strategies. We then evaluated each of these strategies, along with the CFR reference strategy, by running FBR-n with a variety of n values to produce counter-strategies. Figure 5.5 shows the results. Each line corresponds to one of the CFR-FBR-n strategies being tested with the exception of one line which represents the CFR benchmark strategy. The x-axis shows hows many sample iterations were used in FBR to create a counter-strategy to a given test strategy. 95% confidence intervals are not shown, but are on the order of ±5 mbb/g for all values.

For CFR-FBR-n strategies trained with low n values, we see results similar to CFR-UCT results in Leduc. For example, the CFR-UCT-10k strategy outperforms the CFR strategy when each plays against the response strategy it was trained to beat (FBR-10k), but it also has stronger performance when played against FBR-3k or FBR-1k. Thus by training against a FBR-n_1 opponent, the CFR-FBR-n strategies also learn how to play well against FBR-n_2 opponents where $n_2 < n_1$. Even in large games, for at least some interesting response functions f, CFR-f can be used to learn strategies that are strong according to v_f.

On the other hand, CFR-FBR-n does not create strong strategies for higher n values. The CFR-FBR-30k strategy only gets a value approximately equal to the CFR strategy against FBR-30k, and the CFR-FBR-100k strategy does worse than the CFR strategy against FBR-100k. Despite this, the performance of these strategies against FBR-1M show that they are still quite exploitable, so they must differ greatly from the unexploitable CFR strategy. However, there is some evidence that the CFR-FBR-n strategies are still improving; we ran CFR-FBR-100k for approximately 5.8 million CFR iterations, and over the last 1 million iterations its value against FBR-100k improved by approximately 23 mbb/g. Thus, it is possible that if CFR-FBR-n is run for long enough, it will produce strategies that outperform a CFR strategy against FBR-n, even for large n. On the other hand, if CFR-FBR-n never outperforms CFR against FBR-n for large n, it simply shows that there are f for which CFR-f doesn't converge to a near-optimal strategy, which reinforces the importance of the PGR framework for ensuring that CFR-f converges.

Chapter 6

Conclusion

In Chapter 3, we demonstrated that the exploitability metric leaves much to be desired as a measure of strategy strength. By using other metrics to complement it, we can measure not just a strategy's worst-case performance, but how likely it is that an opponent can effectively exploit it. The pretty-good response condition allows us to create evaluation metrics which use adaptive opponents which exploit a strategy but with varying degrees of exploitative power. By using several metrics from this framework, including exploitability, we can capture different dimensions of strategy strength. We demonstrated this empirically in our comparison of a CFR strategy and a CFR-BR strategy. CFR-BR is the clear winner by exploitability, but weaker PGRs measured CFR as the stronger strategy, indicating that CFR strategies are harder to exploit.

In Chapter 4 we introduced a new learning method for any multiagent domain with regret minimization algorithms; in particular, CFR-f is an algorithm for learning with adaptive opponents in extensive-form games. We showed that CFR-f always converges to an optimal strategy when f is a PGR. This means that no matter which PGR we decide to use as an evaluation metric, we can efficiently find a strategy that is optimal under the metric. In addition, we showed that the degree to which we can learn to play against an adaptive opponent is directly related to the degree to which his response function deviates from a PGR.

In Section 4.3, we generalized the idea of Restricted Nash Responses using mixtures of PGRs. This allows us to efficiently combine evaluation metrics when learning a strategy. In particular, by having the opponent mix between any PGR and a regret-minimization player (i.e. CFR), we can search for PGR-optimal strategies within the set of strategies with bounded exploitability. By mixing between a PGR and a static strategy, we can compute different kinds of robust responses. Where the original technique found best-responses within the set of strategies with low exploitability, we can find best-responses within the set of strategies that are hard to exploit.

In Chapter 5 we tested how CFR-f worked in poker games for learning against adaptive opponents. With Monte Carlo responses, we showed how we can use sampling during solving to learn how to play well against opponents who will respond to actual observations of our strategy during online play. The results with UCT in Leduc Hold'em were especially promising, showing that we can find static strategies that do better than an equilibrium against these adaptive opponents. Although we evaluated our strategies by letting UCT learn offline and then only measuring the performance with the final strategy produced by UCT, we also found that our CFR-UCT strategies did well when we played against UCT opponents that use less samples than the training opponent. This means that our CFR-UCT strategies would also do well against an opponent who uses UCT to learn online.

6.1 Future Work

Although we hope that we provided broad answers to the questions of strategy evaluation and learning with new metrics, there is still work to be done. We showed how pretty-good responses are a powerful framework for evaluating strategies, but the question still remains as to how this framework should best be used. There are an uncountably infinite number of possible pretty-good responses, but realistically we can only select a small subset with which to evaluate strategies. Investigating how this subset should be chosen is an interesting line of further work. This isn't just a matter of empirically testing PGRs; by investigating the similarities and differences in how PGRs evaluate strategies, we might be able to find representative PGRs which best evaluate different dimensions of strategy strength.

Although we have shown that CFR-f is an efficient algorithm for learning against response functions, this efficiency is dependent on the computational complexity of f. We know that best-responses, one example of a PGR, are computationally expensive and thus intractable in very large games. In games where the best-response is computable, CFR-BR takes longer to converge than the standard CFR algorithm [25], and this is without considering the gains in efficiency that sampling can give when used in CFR. In order for CFR-f to be a competitive algorithm in large games, we must find a way to make it faster. An obvious way to do this is to use an f that can be quickly calculated, but it is not clear if response functions exist that are both computationally simple and interesting as evaluation tools.

A different approach to the efficiency issue could be to change how f is applied. In the CFR-f algorithm presented here, we calculate $f(\sigma_1^t)$ from scratch on every iteration. For convergence guarantees to hold, though, all we need is for the CFR player to observe utilities *as if* the response player used $f(\sigma_1^t)$ on each iteration. It could be possible to maintain a convergence guarantee while only partially generating the response player's strategy, or while only applying an incremental update on each iteration. For one specific example of how this can work, consider CFR-BR where the CFR agent uses some sampling scheme. In subgames which aren't sampled, we know that the CFR agent's strategy will not change; this also means that the BR player's strategy in these subgames doesn't have to change. Thus we don't have to apply a full best-response calculation on every iteration, but can get away with only updating the sampled subgames.

Bibliography

[1] Annual Computer Poker Competition. `http://www.computerpokercompetition.org/`. Accessed 2015-03-05.

[2] Peter Auer, Nicolo Ces-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. **Machine Learning**, 47(2-3):235–256, 2002.

[3] Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modeling. In **Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2013.

[4] Nolan Bard, Michael Johanson, and Michael Bowling. Asymmetricabstraction for adversarial settings. In **Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2014.

[5] David Blackwell. An analog of the minimax theorem for vector payoffs. **Pacific Journal of Mathematics**, 6(1):1–8, 1956.

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

[7] Noam Brown, Sam Ganzfried, and Tuomas Sandholm. Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit texas hold'em agent. In **Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2015. to appear.

[8] Neil Burch. Private communication, 2015.

[9] Trevor Davis, Neil Burch, and Michael Bowling. Using response functions to measure strategy strength. In **Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)**, 2014.

[10] Yoav Freund and Robert Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Compute and Science Systems**, 55(1):119–139, 1997.

[11] Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In **Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2011.

[12] Sam Ganzfried and Tuomas Sandholm. Action translation in extensive-form games with large action spaces: Axioms, paradoxes, and the pseudo-harmonic mapping. In **Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)**, 2013 (to appear).

[13] Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect information games. In **Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)**, 2014.

[14] Sam Ganzfried, Tuomas Sandholm, and Kevin Waugh. Strategy purification and thresholding: Effective non-equilibrium approaches for playing large games. In **Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2012.

[15] Sylvain Gelly and Yizao Wang. Exploration exploitation in Go: UCT for Monte-Carlo Go. In **Advances in Neural Information Processing Systems 19 (NIPS)**, 2006.

[16] Richard Gibson, Marc Lanctot, Neil Burch, Duane Szafron, and Michael Bowling. Generalized sampling and variance in counterfactual regret minimization. In **Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)**, 2012.

[17] Andrew Gilpin and Tuomas Sandholm. Expectation-based versus potential-aware automated abstraction in imperfect information games: An experimental comparison using poker. In **Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)**, 2008.

[18] Geoffrey Gordon. No-regret algorithms for online convex programs. In **Advances in Neural Information Processing Systems 19 (NIPS)**, 2006.

[19] Amy Greenwald, Zheng Li, and Casey Marks. Bounds for regret-matching algorithms. Technical report, Brown University, 2006.

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

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

[22] Michael Johanson. Measuring the size of large no-limit poker games. Technical Report TR13-01, Department of Computing Science, University of Alberta, 2013.

[23] Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In **Advances in Neural Information Processing Systems 20 (NIPS)**, 2007.

[24] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In **Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)**, 2011.

[25] Michael Johanson, Nolan Bard, Neil Burch, and Michael Bowling. Finding optimal abstract strategies in extensive-form games. In **Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)**, 2012.

[26] Michael Johanson, Nolan Bard, Marc Lanctot, Richard Gibson, and Michael Bowling. Efficient nash equilibrium approximation through monte carlo counterfactual regret minimization. In **Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2012.

[27] Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In **Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2013.

[28] Levente Kocsis and Csaba Szepsvari. Bandit based Monte-Carlo planning. In **Proceedings of the 17th European Conference on Machine Learning (ECML)**, 2006.

[29] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In **Advances in Neural Information Processing Systems 22 (NIPS)**, 2009.

[30] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. **Information and Computing**, 108(2):212–261, 1994.

[31] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In **Proceedings of the Twenty-First Conference on Uncertainty in Artficial Intelligence (UAI)**, 2005.

[32] Kevin Waugh. Abstraction in large extensive games. Master's thesis, University of Alberta, Edmonton, Alberta, Canada, 2009.

[33] Kevin Waugh, David Schnizlein, Michael Bowling, and Duane Szafron. Abstraction pathologies in extensive games. In **Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)**, 2009.

[34] Kevin Waugh, Martin Zinkevich, Michael Johanson, Morgan Kan, David Schnizlein, and Michael Bowling. A practical use of imperfect recall. In **Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation (SARA)**, 2009.

[35] Kevin Waugh, Dustin Morrill, J. Andrew Bagnell, and Michael Bowling. Solving games with functional regret estimation. In **Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)**, 2015.

[36] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In **Advances in Neural Information Processing Systems 20 (NIPS)**, 2007.