Regret Minimization in Games with Incomplete Information

Martin Zinkevich
maz@cs.ualberta.ca

Michael Johanson
johanson@cs.ualberta.ca

Michael Bowling
Computing Science Department
University of Alberta
Edmonton, AB Canada T6G2E8
bowling@cs.ualberta.ca

Carmelo Piccione
Computing Science Department
University of Alberta
Edmonton, AB Canada T6G2E8
carm@cs.ualberta.ca

Abstract

Extensive games are a powerful model of multiagent decision-making scenarios with incomplete information. Finding a Nash equilibrium for very large instances of these games has received a great deal of recent attention. In this paper, we describe a new technique for solving large games based on regret minimization. In particular, we introduce the notion of counterfactual regret, which exploits the degree of incomplete information in an extensive game. We show how minimizing counterfactual regret minimizes overall regret, and therefore in self-play can be used to compute a Nash equilibrium. We demonstrate this technique in the domain of poker, showing we can solve abstractions of limit Texas Hold'em with as many as 10^{12} states, two orders of magnitude larger than previous methods.

1 Introduction

Extensive games are a natural model for sequential decision-making in the presence of other decision-makers, particularly in situations of imperfect information, where the decision-makers have differing information about the state of the game. As with other models (e.g., MDPs and POMDPs), its usefulness depends on the ability of solution techniques to scale well in the size of the model. Solution techniques for very large extensive games have received considerable attention recently, with poker becoming a common measuring stick for performance. Poker games can be modeled very naturally as an extensive game, with even small variants, such as two-player, limit Texas Hold'em, being impractically large with just under 10^{18} game states.

State of the art in solving extensive games has traditionally made use of linear programming using a realization plan representation [1]. The representation is linear in the number of game states, rather than exponential, but considerable additional technology is still needed to handle games the size of poker. Abstraction, both hand-chosen [2] and automated [3], is commonly employed to reduce the game from 10^{18} to a tractable number of game states (e.g., 10^7), while still producing strong poker programs. In addition, dividing the game into multiple subgames each solved independently or in real-time has also been explored [2, 4]. Solving larger abstractions yields better approximate Nash equilibria in the original game, making techniques for solving larger games the focus of research in this area. Recent iterative techniques have been proposed as an alternative to the traditional linear programming methods. These techniques have been shown capable of finding approximate solutions to abstractions with as many as 10^{10} game states [5, 6, 7], resulting in the first significant improvement in poker programs in the past four years.

In this paper we describe a new technique for finding approximate solutions to large extensive games. The technique is based on regret minimization, using a new concept called counterfactual regret. We show that minimizing counterfactual regret minimizes overall regret, and therefore can be used to compute a Nash equilibrium. We then present an algorithm for minimizing counterfactual regret in poker. We use the algorithm to solve poker abstractions with as many as 10^{12} game states, two orders of magnitude larger than previous methods. We also show that this translates directly into an improvement in the strength of the resulting poker playing programs. We begin with a formal description of extensive games followed by an overview of regret minimization and its connections to Nash equilibria.

2 Extensive Games, Nash Equilibria, and Regret

Extensive games provide a general yet compact model of multiagent interaction, which explicitly represents the often sequential nature of these interactions. Before presenting the formal definition, we first give some intuitions. The core of an extensive game is a game tree just as in perfect information games (e.g., Chess or Go). Each non-terminal game state has an associated player choosing actions and every terminal state has associated payoffs for each of the players. The key difference is the additional constraint of information sets, which are sets of game states that the controlling player cannot distinguish and so must choose actions for all such states with the same distribution. In poker, for example, the first player to act does not know which cards the other players were dealt, and so all game states immediately following the deal where the first player holds the same cards would be in the same information set. We now describe the formal model as well as notation that will be useful later.

Definition 1 *[8, p. 200] a finite extensive game with imperfect information has the following components:*

- *A finite set N of* ***players****.*
- *A finite set H of sequences, the possible* ***histories*** *of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the terminal histories (those which are not a prefix of any other sequences). $A(h) = \{a : (h, a) \in H\}$ are the actions available after a nonterminal history $h \in H$,*
- *A function P that assigns to each nonterminal history (each member of $H \backslash Z$) a member of $N \cup \{c\}$. P is the* ***player function****. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$ then chance determines the action taken after history h.*
- *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*
- *For each player $i \in N$ a partition $\mathcal{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. For $I_i \in \mathcal{I}_i$ we denote by $A(I_i)$ the set $A(h)$ and by $P(I_i)$ the player $P(h)$ for any $h \in I_i$. $\mathcal{I}_i$ is the* ***information partition*** *of player i; a set $I_i \in \mathcal{I}_i$ is an* ***information set*** *of player i.*
- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals $\mathbf{R}$. If $N = \{1, 2\}$ and $u_1 = -u_2$, it is a* ***zero-sum extensive game****. Define $\Delta_{u,i} = \max_z u_i(z) - \min_z u_i(z)$ to be the range of utilities to player i.*

Note that the partitions of information as described can result in some odd and unrealistic situations where a player is forced to forget her own past decisions. If all players can recall their previous actions and the corresponding information sets, the game is said to be one of **perfect recall**. This work will focus on finite, zero-sum extensive games with perfect recall.

2.1 Strategies

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

Let $\pi^\sigma(h)$ be the probability of history h occurring if players choose actions according to σ. We can decompose $\pi^\sigma = \Pi_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ into each player's contribution to this probability. Hence, $\pi_i^\sigma(h)$ is the probability that if player i plays according to σ then for all histories h' that are a proper prefix of h with $P(h') = i$, player i takes the corresponding action in h. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except player i. For $I \subseteq H$, define $\pi^\sigma(I) = \sum_{h \in I} \pi^\sigma(h)$, as the probability of reaching a particular information set given σ, with $\pi_i^\sigma(I)$ and $\pi_{-i}^\sigma(I)$ defined similarly.

The overall value to player i of a strategy profile is then the expected payoff of the resulting terminal node, $u_i(\sigma) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$.

2.2 Nash Equilibrium

The traditional solution concept of a two-player extensive game is that of a Nash equilibrium. A **Nash equilibrium** is a strategy profile σ where

$$u_1(\sigma) \geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \qquad\qquad u_2(\sigma) \geq \max_{\sigma_2' \in \Sigma_2} u_2(\sigma_1, \sigma_2'). \tag{1}$$

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

$$u_1(\sigma) + \epsilon \geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \qquad\qquad u_2(\sigma) + \epsilon \geq \max_{\sigma_2' \in \Sigma_2} u_2(\sigma_1, \sigma_2'). \tag{2}$$

2.3 Regret Minimization

Regret is an online learning concept that has triggered a family of powerful learning algorithms. To define this concept, first consider repeatedly playing an extensive game. Let σ_i^t be the strategy used by player i on round t. The **average overall regret** of player i at time T is:

$$R_i^T = \frac{1}{T} \max_{\sigma_i^* \in \Sigma_i} \sum_{t=1}^{T} \left(u_i(\sigma_i^*, \sigma_{-i}^t) - u_i(\sigma^t) \right) \tag{3}$$

Moreover, define $\bar{\sigma}_i^t$ to be the average strategy for player i from time 1 to T. In particular, for each information set $I \in \mathcal{I}_i$, for each $a \in A(I)$, define:

$$\bar{\sigma}_i^t(I)(a) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(I)(a)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}. \tag{4}$$

There is a well-known connection between regret and the Nash equilibrium solution concept.

Theorem 2 *In a zero-sum game at time T, if both player's average overall regret is less than ϵ, then $\bar{\sigma}^T$ is a 2ϵ equilibrium.*

An algorithm for selecting σ_i^t for player i is regret minimizing if player i's average overall regret (regardless of the sequence σ_{-i}^t) goes to zero as t goes to infinity. As a result, regret minimizing algorithms in self-play can be used as a technique for computing an approximate Nash equilibrium. Moreover, an algorithm's bounds on the average overall regret bounds the rate of convergence of the approximation.

Traditionally, regret minimization has focused on bandit problems more akin to normal-form games. Although it is conceptually possible to convert any finite extensive game to an equivalent normal-form game, the exponential increase in the size of the representation makes the use of regret algorithms on the resulting game impractical. Recently, Gordon has introduced the Lagrangian Hedging (LH) family of algorithms, which can be used to minimize regret in extensive games by working with the realization plan representation [5]. We also propose a regret minimization procedure that exploits the compactness of the extensive game. However, our technique doesn't require the costly quadratic programming optimization needed with LH allowing it to scale more easily, while achieving even tighter regret bounds.

3 Counterfactual Regret

The fundamental idea of our approach is to decompose overall regret into a set of additive regret terms, which can be minimized independently. In particular, we introduce a new regret concept for extensive games called counterfactual regret, which is defined on a individual information set. We show that overall regret is bounded by the sum of counterfactual regret, and also show how counterfactual regret can be minimized at each information set independently.

We begin by considering one particular information set $I \in \mathcal{I}_i$ and player i's choices made in that information set. Define $u_i(\sigma, h)$ to be the expected utility given that the history h is reached and then all players play using strategy σ. Define **counterfactual utility** $u(\sigma, I)$ to be the expected utility given that information set I is reached and all players play using strategy σ except that player i plays to reach I. This can be thought of as a "counterfactual" as it is the value to player i of reaching information set I *if the player had tried to do so.* Finally, for all $a \in A(I)$, define $\sigma|_{I\to a}$ to be a strategy profile identical to σ except that player i always chooses action a when in information set I. The **immediate counterfactual regret** is:

$$R^T_{i,\mathrm{imm}}(I) = \frac{1}{T} \max_{a\in A(I)} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I)\right) \tag{5}$$

Intuitively, this is the player's regret in its decisions at information set I in terms of counterfactual utility, with an additional weighting term for the counterfactual probability that I would be reached on that round *if the player had tried to do so.* As we will often be most concerned about regret when it is positive, let $R^{T,+}_{i,\mathrm{imm}}(I) = \max(R^T_{i,\mathrm{imm}}(I), 0)$ be the positive portion of immediate counterfactual regret.

We can now state our first key result.

Theorem 3 $R^T_i \le \sum_{I\in\mathcal{I}_i} R^{T,+}_{i,\mathrm{imm}}(I)$

This proof is in the appendix[1]. Since minimizing immediate counterfactual regret minimizes the overall regret, it enables us to find an approximate Nash equilibrium if we can only minimize the immediate counterfactual regret.

The key feature of immediate counterfactual regret is that it can be minimized by controlling only $\sigma_i(I)$. To this end, we can use Blackwell's algorithm for approachability to minimize this regret independently on each information set. In particular, we maintain for all $I \in \mathcal{I}_i$, for all $a \in A(I)$:

$$R^T_i(I, a) = \frac{1}{T} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I)\right) \tag{6}$$

Define $R^{T,+}_i(I, a) = \max(R^T_i(I, a), 0)$, then the strategy for time $T + 1$ is:

$$\sigma^{T+1}_i(I)(a) = \begin{cases} \frac{R^{T,+}_i(I,a)}{\sum_{a\in A(I)} R^{T,+}_i(I,a)} & \text{if } \sum_{a\in A(I)} R^{T,+}_i(I, a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \tag{7}$$

In other words, actions are selected in proportion to the amount of positive counterfactual regret for not playing that action. If no actions have any positive counterfactual regret, then the action is selected randomly. This leads us to our second key result.

Theorem 4 *If player i selects actions according to Equation 7 then $R^T_{i,\mathrm{imm}}(I) \le \Delta_{u,i}\sqrt{|A_i|}/\sqrt{T}$ and consequently $R^T_i \le \Delta_{u,i}|\mathcal{I}_i|\sqrt{|A_i|}/\sqrt{T}$ where $|A_i| = \max_{h:P(h)=i} |A(h)|$.*

The proof is in the appendix. This result establishes that the strategy in Equation 7 can be used in self-play to compute a Nash equilibrium. In addition, the bound on the average overall regret is linear in the number of information sets. These are similar bounds to what's achievable by Gordon's Lagrangian Hedging algorithms. Meanwhile, minimizing counterfactual regret does not require a costly quadratic program projection on each iteration. In the next section we demonstrate our technique in the domain of poker.

[1]Material noted as being in the appendix is available in the supporting material accompanying this submission. It will be made available in a companion technical report if this paper is accepted for publication.

4 Application To Poker

We now describe how we use counterfactual regret minimization to compute a near equilibrium solution in the domain of poker. The poker variant we focus on is heads-up limit Texas Hold'em, as it is used in the AAAI Computer Poker Competition[9]. The game consists of two players (zero-sum), four rounds of cards being dealt, and four rounds of betting, and has just under 10^{18} game states [2]. As with all previous work on this domain, we will first abstract the game and find an equilibrium of the abstracted game. In the terminology of extensive games, we will merge information sets; in the terminology of poker, we will bucket card sequences. The quality of the resulting near equilibrium solution depends on the coarseness of the abstraction. In general, the less abstraction used, the higher the quality of the resulting strategy. Hence, the ability to solve a larger game means less abstraction is required, translating into a stronger poker playing program.

4.1 Abstraction

The goal of abstraction is to reduce the number of information sets for each player to a tractable size such that the abstract game can be solved. Early poker abstractions [2, 4] involved limiting the possible sequences of bets, e.g., only allowing three bets per round, or replacing all first-round decisions with a fixed policy. More recently, abstractions involving full four round games with the full four bets per round have proven to be a significant improvement [7, 6]. We also will keep the full game's betting structure and focus abstraction on the dealt cards.

Our abstraction groups together observed card sequences based on a metric called hand strength squared. Hand strength is the expected probability of winning[2] given only the cards a player has seen. This was used a great deal in previous work on abstraction [2, 4]. Hand strength squared is the expected square of the hand strength after the last card is revealed, given only the cards a player has seen. Intuitively, hand strength squared is similar to hand strength but gives a bonus to card sequences whose eventual hand strength has higher variance. Higher variance is preferred as it means the player eventually will be more certain about their ultimate chances of winning prior to a showdown. More importantly, we will show in Section 5 that this metric for abstraction results in stronger poker strategies.

The final abstraction is generated by partitioning card sequences based on the hand strength squared metric. First, all round-one card sequences (i.e., all private card holdings) are partitioned into ten equally sized buckets based upon the metric. Then, all round-two card sequences *that shared a round-one bucket* are partitioned into ten equally sized buckets based on the metric now applied at round two. Thus, a partition of card sequences in round two is a pair of numbers: its bucket in the previous round and its bucket in the current round given its bucket in the previous round. This is repeated after reach round, continuing to partition card sequences that agreed on the previous rounds' buckets into ten equally sized buckets based on the metric applied in that round. Thus, card sequences are partitioned into **bucket sequences**: a bucket from $\{1, \ldots 10\}$ for each round. The resulting abstract game has approximately 1.65×10^{12} game states, and 5.73×10^{7} information sets. In the full game of poker, there are approximately 9.17×10^{17} game states and 3.19×10^{14} information sets. So although this represents a significant abstraction on the original game it is two orders of magnitude larger than previously solved abstractions.

4.2 Minimizing Counterfactual Regret

Now that we have specified an abstraction, we can use counterfactual regret minimization to compute an approximate equilibrium for this game. The basic procedure involves having two players repeatedly play the game using the counterfactual regret minimizing strategy from Equation 7. After T repetitions of the game, or simply *iterations*, we return $(\bar{\sigma}_1^T, \bar{\sigma}_2^T)$ as the resulting approximate equilibrium. Repeated play requires storing $R_i^t(I, a)$ for every information set I and action a, and updating it after each iteration.[3]

[2]Where a tie is considered "half a win"

[3]The bound from Theorem 4 for the basic procedure can actually be made significantly tighter in the specific case of poker. In the appendix, we show that the bound for poker is actually independent of the size of the card abstraction.

For our experiments, we actually use a variation of this basic procedure, which exploits the fact that our abstraction has a small number of information sets relative to the number of game states. Although each information set is crucial, many consist of a hundred or more individual histories. This fact suggests it may be possible to get a good idea of the correct behavior for an information set by only sampling a fraction of the associated game states. In particular, for each iteration, we sample deterministic actions for the chance player. Thus, σ_c^t is set to be a deterministic strategy, but chosen according to the distribution specified by f_c. For our abstraction this amounts to choosing a joint bucket sequence for the two players. Once the joint bucket sequence is specified, there are only 18,496 reachable states and 6,378 reachable information sets. Since $\pi_{-i}^{\sigma^t}(I)$ is zero for all other information sets, no updates need to be made for these information sets.[4]

This sampling variant allows approximately 750 iterations of the algorithm to be completed in a single second on a single core of a 2.4Ghz Dual Core AMD Opteron 280 processor. In addition, a straightforward parallelization is possible and was used when noted in the experiments. Since betting is public information, the flop-onward information sets for a particular preflop betting sequence can be computed independently. With four processors we were able to complete approximately 1700 iterations in one second. The complete algorithmic details with pseudocode can be found in the appendix.

5 Experimental Results

Before discussing the results, it is useful to consider how one evaluates the strength of a near equilibrium poker strategy. One natural method is to measure the strategy's exploitability, or its performance against its worst-case opponent. In a symmetric, zero-sum game like heads-up poker[5], a perfect equilibrium has zero exploitability, while an ϵ-Nash equilibrium has exploitability ϵ. A convenient measure of exploitability is millibets-per-hand (mb/h), where a millibet is one thousandth of a small-bet, the fixed magnitude of bets used in the first two rounds of betting. To provide some intuition for these numbers, a player that always folds will lose 750 mb/h while a player that is 10 mb/h stronger than another would require over one million hands to be 95% certain to have won overall.

In general, it is intractable to compute a strategy's exploitability within the full game. For strategies in a reasonably sized abstraction it is possible to compute their exploitability within their own abstract game. Such a measure is a useful evaluation of the equilibrium computation technique that was used to generate the strategy. However, it does not imply the technique cannot be exploited by a strategy outside of its abstraction. It is therefore common to compare the performance of the strategy in the full game against a battery of known strong poker playing programs. Although positive expected value against an opponent is not transitive, winning against a large and diverse range of opponents suggests a strong program.

We used the sampled counterfactual regret minimization procedure to find an approximate equilibrium for our abstract game as described in the previous section. The algorithm was run for 2 billion iterations ($T = 2 \times 10^9$), or less than 14 days of computation when parallelized across four CPUs. The resulting strategy's exploitability within its own abstract game is 2.2 mb/h. After only 200 million iterations, or less than 2 days of computation, the strategy was already exploitable by less than 13 mb/h. Notice that the algorithm visits only 18,496 game states per iteration. After 200 million iterations each game state has been visited less than 2.5 times on average, yet the algorithm has already computed a relatively accurate solution.

5.1 Scaling the Abstraction

In addition to finding an approximate equilibrium for our large abstraction, we also found approximate equilibria for a number of smaller abstractions. These abstractions used fewer buckets per round to partition the card sequences. In addition to ten buckets, we also solved eight, six, and five

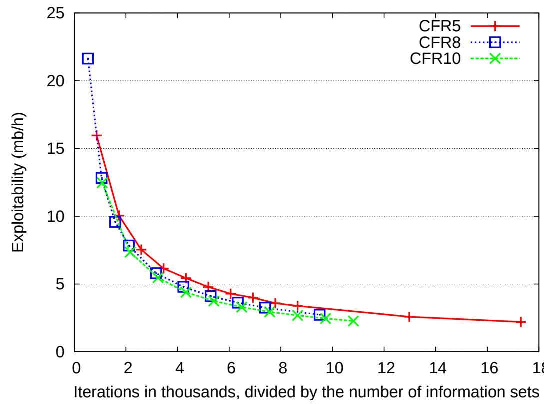
[4]A regret analysis of this variant in poker is included in the appendix. We show that the quadratic decrease in the cost per iteration only causes in a linear increase in the required number of iterations. The experimental results in the next section coincides with this analysis.

[5]A single hand of poker is not a symmetric game as the order of betting is strategically significant. However a pair of hands where the betting order is reversed is symmetric.

Abs	Size ($\times 10^9$)	Iterations ($\times 10^6$)	Time (h)	Exp (mb/h)
5	6.45	100	33	3.4
6	27.7	200	75	3.1
8	276	750	261	2.7
10	1646	2000	326†	2.2

†: parallel implementation with 4 CPUs

(a)



(b)

Figure 1: (a) Number of game states, number of iterations, computation time, and exploitability (in its own abstract game) of the resulting strategy for different sized abstractions. (b) Convergence rates for three different sized abstractions. The x-axis shows the number of iterations divided by the number of information sets in the abstraction.

bucket variants. As these abstractions are smaller, they require fewer iterations to compute a similarly accurate equilibrium. For example, the program computed with the five bucket approximation (CFR5) is about 250 times smaller with just under 10^{10} game states. After 100 million iterations, or 33 hours of computation without any parallelization, the final strategy is exploitable by 3.4 mb/h. This is approximately the same size of game solved by recent state-of-the-art algorithms [6, 7] with many days of computation.

Figure 1b shows a graph of the convergence rates for the five, eight, and ten partition abstractions. The y-axis is exploitability while the x-axis is the number of iterations normalized by the number of information sets in the particular abstraction being plotted. The rates of convergence almost exactly coincide showing that, in practice, the number of iterations needed is growing linearly with the number of information sets. Due to the use of sampled bucket sequences, the time per iteration is nearly independent of the size of the abstraction. This suggests that, in practice, the overall computational complexity is only linear in the size of the chosen card abstraction.

5.2 Performance in Full Texas Hold'em

We have noted that the ability to solve larger games means less abstraction is necessary, resulting in an overall stronger poker playing program. We have played our four near equilibrium bots with various abstraction sizes against each other and two other known strong programs: PsOpti4 and S2298. PsOpti4 is a variant of the equilibrium strategy described in [2]. It was the stronger half of Hyperborean, the AAAI 2006 Computer Poker Competition's winning program. It is available under the name SparBot in the entertainment program Poker Academy, published by BioTools. We have calculated strategies that exploit it at 175 mb/h. S2298 is the equilibrium strategy described in [6]. We have calculated strategies that exploit it at 52.5 mb/h. In terms of the size of the abstract game PsOpti4 is the smallest consisting of a small number of merged three round games. S2298 restricts the number of bets per round to 3 and uses a five bucket per round card abstraction based on hand-strength, resulting an abstraction slightly smaller than CFR5.

Table 1 shows a cross table with the results of these matches. Strategies from larger abstractions consistently, and significantly, outperform their smaller counterparts. The larger abstractions also consistently exploit weaker bots by a larger margin (e.g., CFR10 wins 19mb/h more from S2298 than CFR5).

Finally, we also played CFR8 against the four bots that competed in the bankroll portion of the 2006 AAAI Computer Poker Competition, which are available on the competition's benchmark server [9]. The results are shown in Table 2, along with S2298's previously published performance against the

	PsOpti4	S2298	CFR5	CFR6	CFR8	CFR10	Average
PsOpti4	0	-28	-36	-40	-52	-55	-35
S2298	28	0	-17	-24	-30	-36	-13
CFR5	36	17	0	-5	-13	-20	2
CFR6	40	24	5	0	-9	-14	7
CFR8	52	30	13	9	0	-6	16
CFR10	55	36	20	14	6	0	22
Max	55	36	20	14	6	0	

Table 1: Winnings in mb/h for the row player in full Texas Hold'em. Matches with Opti4 used 10 duplicate matches of 10,000 hands each and are significant to 20 mb/h. Other matches used 10 duplicate matches of 500,000 hands each are are significant to 2 mb/h.

	Hyperborean	BluffBot	Monash	Teddy	Average
S2298	61	113	695	474	336
CFR8	106	170	746	517	385

Table 2: Winnings in mb/h for the row player in full Texas Hold'em.

same bots [6]. The program not only beats all of the bots from the competition but does so by a larger margin than S2298.

6 Conclusion

We introduced a new regret concept for extensive games called counterfactual regret. We showed that minimizing counterfactual regret minimizes overall regret and presented a general and poker-specific algorithm for efficiently minimizing counterfactual regret. We demonstrated the technique in the domain of poker, showing that the technique can compute an approximate equilibrium for abstractions with as many as 10^{12} states, two orders of magnitude larger than previous methods. We also showed that the resulting poker playing program outperforms other strong programs, including all of the competitors from the bankroll portion of the 2006 AAAI Computer Poker Competition.

References

[1] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, pages 528–552, 1992.

[2] D. Billings, N. Burch, A. Davidson, R. Holte, J. Schaeffer, T. Schauenberg, and D. Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *International Joint Conference on Artificial Intelligence*, pages 661–668, 2003.

[3] A. Gilpin and T. Sandholm. Finding equilibria in large sequential games of imperfect information. In *ACM Conference on Electronic Commerce*, 2006.

[4] A. Gilpin and T. Sandholm. A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In *National Conference on Artificial Intelligence*, 2006.

[5] G. Gordon. No-regret algorithms for online convex programs. In *Neural Information Processing Systems 19*, 2007.

[6] M. Zinkevich, M. Bowling, and N. Burch. A new algorithm for generating strong strategies in massive zero-sum games. In *Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI)*, 2007. To Appear.

[7] A. Gilpin, S. Hoda, J. Pena, and T. Sandholm. Gradient-based algorithms for finding nash equilibria in extensive form games. In *Proceedings of the Eighteenth International Conference on Game Theory*, 2007.

[8] M. Osborne and A. Rubenstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[9] M. Zinkevich and M. Littman. The AAAI computer poker competition. *Journal of the International Computer Games Association*, 29, 2006. News item.

A Appendix

A.1 Proof of Theorem 3

Define $D(I)$ to be the information sets of player i reachable from I (including I). Define $\sigma|_{D(I)\to\sigma'}$ to be a strategy profile equal to σ except in the information sets in $D(I)$ where it is equal to σ'. The **full counterfactual regret** is:

$$R^T_{i,\text{full}}(I) = \frac{1}{T} \max_{\sigma' \in \Sigma_1} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \left(u_i(\sigma^t|_{D(I)\to\sigma'}, I) - u_i(\sigma^t, I) \right) \tag{8}$$

Again, we define $R^{T,+}_{1,\text{full}}(I) = \max(R^T_{1,\text{full}}(I), 0)$. Moreover, we define $\text{succ}^\sigma_i(I'|I,a)$ to be the probability that I' is the next information set of player i visited given that the action a was just selected in information set I, and σ is the current strategy. If σ implies that I is unreachable because of an action of player i, that action is changed to allow I to be reachable. Define $\text{Succ}_i(I,a)$ to be the set of all possible next information sets of player i visited given that action $a \in A(I)$ was just selected in information set I. Define $\text{Succ}_i(I) = \bigcup_{a\in A(I)} \text{Succ}_i(I,a)$.

The following lemma describes the relationship between full and immediate counterfactual regret.

Lemma 5 $R^T_{i,\text{full}}(I) \le R^T_{i,\text{imm}}(I) + \sum_{I'\in\text{Succ}_i(I)} R^{T,+}_{i,\text{full}}(I')$

Proof:

$$\begin{aligned} R^T_{i,\text{full}}(I) = {}& \frac{1}{T} \max_{a\in A(I)} \max_{\sigma'\in\Sigma_1} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \\ & \left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I) + \sum_{I'\in\text{Succ}_i(I,a)} \text{succ}^\sigma_i(I'|I,a) \left(u_i(\sigma^t|_{(D(I)\to\sigma'}, I') - u_i(\sigma^t, I') \right) \right) \end{aligned} \tag{9}$$

$$\begin{aligned} R^T_{i,\text{full}}(I) \le {}& \frac{1}{T} \max_{a\in A(I)} \max_{\sigma'\in\Sigma_1} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I) \right) \\ & + \frac{1}{T} \max_{a\in A(I)} \max_{\sigma'\in\Sigma_1} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I) \sum_{I'\in\text{Succ}_i(I,a)} \text{succ}^\sigma_i(I'|I,a) \left(u_i(\sigma^t|_{(D(I)\to\sigma'}, I') - u_i(\sigma^t, I') \right) \end{aligned} \tag{10}$$

The first part of the expression on the right hand side is the immediate regret. For the second, we know that $\pi^{\sigma^t}_{-i}(I)\text{succ}^\sigma_i(I'|I,a) = \pi^{\sigma^t}_{-i}(I')$, and that $u_i(\sigma^t|_{D(I)\to\sigma'}, I') = u_i(\sigma^t|_{D(I')\to\sigma'}, I')$.

$$\begin{aligned} R^T_{i,\text{full}}(I) \le {}& R^T_{i,\text{imm}}(I) \\ & + \frac{1}{T} \max_{a\in A(I)} \max_{\sigma'\in\Sigma_1} \sum_{t=1}^{T} \sum_{I'\in\text{Succ}_i(I,a)} \pi^{\sigma^t}_{-i}(I') \left(u_i(\sigma^t|_{(D(I')\to\sigma'}, I') - u_i(\sigma^t, I') \right) \end{aligned} \tag{11}$$

$$\begin{aligned} R^T_{i,\text{full}}(I) \le {}& R^T_{i,\text{imm}}(I) \\ & + \max_{a\in A(I)} \sum_{I'\in\text{Succ}_i(I,a)} \frac{1}{T} \max_{a\in A(I)} \max_{\sigma'\in\Sigma_1} \sum_{t=1}^{T} \pi^{\sigma^t}_{-i}(I') \left(u_i(\sigma^t|_{(D(I')\to\sigma'}, I') - u_i(\sigma^t, I') \right) \end{aligned} \tag{12}$$

$$R^T_{i,\text{full}}(I) \le R^T_{i,\text{imm}}(I) + \max_{a\in A(I)} \sum_{I'\in\text{Succ}_i(I,a)} R^T_{i,\text{full}}(I') \tag{13}$$

Because the game is perfect recall, given distinct $a, a' \in A(I)$, $\text{Succ}_i(I,a)$ and $\text{Succ}_i(I,a')$ are disjoint. If we define, $\text{Succ}_i(I) = \bigcup_{a\in A(I)} \text{Succ}_i(I,a)$, then:

$$R^T_{i,\text{full}}(I) \le R^T_{i,\text{imm}}(I) + \sum_{I'\in\text{Succ}_i(I)} R^{T,+}_{i,\text{full}}(I') \tag{14}$$

■

We prove Theorem 3 by using a lemma that can be proven recursively:

Lemma 6 $R^T_{i,\mathrm{full}}(I) \leq \sum_{I' \in D(I)} R^{T,+}_{i,\mathrm{imm}}(I')$.

Proof: We prove this for a particular game recursively on the size of $D(I)$. Observe that if an information set has no successors, then Lemma 5 proves the result. We use this as a basis step. Also, observe that $D(I) = \{I\} \cup \bigcup_{I' \in \mathrm{Succ}_i(I)} D(I')$, and that if $I' \in \mathrm{Succ}_i(I)$, then $I \notin D(I')$, implying $|D(I')| < |D(I)|$. Thus, by induction we can establish that:

$$R^T_{i,\mathrm{full}}(I) \leq R^T_{i,\mathrm{imm}}(I) + \sum_{I' \in \mathrm{Succ}_i(I)} \sum_{I'' \in \mathrm{Succ}_i(I')} R^{T,+}_{i,\mathrm{imm}}(I'') \tag{15}$$

$$\leq R^{T,+}_{i,\mathrm{imm}}(I) + \sum_{I' \in \mathrm{Succ}_i(I)} \sum_{I'' \in \mathrm{Succ}_i(I')} R^{T,+}_{i,\mathrm{imm}}(I'') \tag{16}$$

Because the game is perfect recall, for any distinct $I', I'' \in \mathrm{Succ}_i(I)$, $D(I')$ and $D(I'')$ are disjoint. Therefore:

$$R^{T,+}_{i,\mathrm{imm}}(I) + \sum_{I' \in \mathrm{Succ}_i(I)} \sum_{I'' \in \mathrm{Succ}_i(I')} R^{T,+}_{i,\mathrm{imm}}(I'') = \sum_{I' \in D(I)} R^{T,+}_{i,\mathrm{imm}}(I') \tag{17}$$

The result immediately follows. ■

Proof (of Theorem 3): If $P(\emptyset) = i$, then $R^T_{i,\mathrm{full}}(\{\emptyset\}) = R^T_i$, and the theorem follows from Lemma 6. If this is not the case, then we can simply add a new information set at the beginning of the game, where player i only has one action. ■

A.2 Regret Matching

Blackwell's approachability theorem when applied to minimizing regret is known as **regret matching**. In general, regret matching can be defined in a domain where there are a fixed set of actions A, a function $u^t : A \to \mathbf{R}$, and on each round a distribution over the actions p^t is selected.

Define the regret of not playing action $a \in A$ until time T as:

$$R^t(a) = \frac{1}{T} \sum_{t=1}^{T} u^t(a) - \sum_{a \in A} p^t(a) u^t(a) \tag{18}$$

and define $R^{t,+}(a) = \max(R^t(a), 0)$. To apply regret matching, one chooses the distribution:

$$p^t(a) = \begin{cases} \frac{R^{t-1,+}(a)}{\sum_{a' \in A} R^{t-1,+}(a')} & \text{if } \sum_{a' \in A} R^{t-1,+}(a') > 0 \\ \frac{1}{|A|} & \text{otherwise} \end{cases} \tag{19}$$

Theorem 7 *If $|u| = \max_{t \in \{1 \ldots T\}} \max_{a, a' \in A} (u^t(a) - u^t(a'))$, the regret of the regret matching algorithm is bounded by:*

$$\max_{a \in A} R^t(a) \leq \frac{|u| \sqrt{|A|}}{\sqrt{T}} \tag{20}$$

Blackwell's original result [**?**] focused on the case where an action (or vector) is chosen at random (instead of a distribution over actions) and gave a probabilistic guarantee. The result above focuses on the distributions selected, and is more applicable to a scenario where a probability is selected instead of an action.

For a proof, see [5].

A.3 Proof of Theorem 4

Observe that Equation 7 is an implementation of regret matching. Moreover, observe that for all $I \in \mathcal{I}_i$, $a \in A(I)$, $\pi^{\sigma^t}_{-i}(u_i(\sigma^t|_{I \to a}, I) - u_i(\sigma^t, I)) \leq \Delta_{u,i}$. Therefore, Theorem 7 states that the counterfactual regret of that node will be less than $\Delta_{u,i}\sqrt{|A(I)|}/\sqrt{T} \leq \Delta_{u,i}|A_i|/\sqrt{T}$. Summing over all $I \in \mathcal{I}_i$ yields the result.

A.4 Poker-Specific Implementation

We need to iterate over all of the information sets reachable given the joint bucket sequence, and compute probabilities and regrets. In order to do this swiftly, we represent the data in each information set in a "player view tree": in other words, we never explicitly represent every state in the abstracted game: instead, we represent the information sets for each player in its own tree, with each node n being one of four types:

- **Bucket Nodes**: nodes representing where information about the cards is observed. Has a child node (an opponent or player node) for each different class that could be observed at that point.
- **Opponent Nodes**: nodes representing where the opponent takes an action. Has a child node for each action.
- **Player Nodes**: nodes representing where the current player takes an action. Contains the average regret with respect to each action, the total probability for each action until this point, and a child node for each action (either an opponent, bucket, or terminal node). There is an implicit information set associated with this node, which we will write as $I(n)$.
- **Terminal Nodes**: nodes where the game ends due to someone folding or a showdown. Given the probability of a win,loss, and tie, has sufficient information to compute an expected utility for the hand given that the node was reached.

Each player observes different pieces of information about the game, and therefore travels to a different part of its tree during the computation. Our algorithm recurses over both trees in a paired fashion. Before we begin, define $u'_i(\sigma, I) = \pi_{-i} u_i(\sigma, I)$. For each node in the trees, there will be a value $u_i(\sigma, n)$ which we use in order to compute the values $u_i(\sigma, I)$ and $u_i(\sigma, I, a)$, which is the expected value given information set I is reached and action a is taken.

Algorithm 1 WALKTREES(r_1, r_2, b, p_1, p_2)

Require: A node r_1 for an information set tree for player 1.
Require: A node r_2 for an information set tree for player 2.
Require: A joint bucket sequence b.
Require: A probability p_1 of player 1 playing to reach the node.
Require: A probability p_2 of player 2 playing to reach the node.
Ensure: The utility $u_i(\sigma, r_i)$ for player 1 and player 2.

if r_1 is a player node (meaning r_2 is an opponent node) **then**
  Compute $\sigma_1(I(r_1))$ according to Equation 7.
  for Each action $a \in A(I(r_1))$ **do**
    Find the associated child of c_1 of r_1 and c_2 of r_2.
    Compute $u_1(\sigma, I(r_1), a)$ and $u_2(\sigma, r_2, a)$ from WALKTREES$(c_1, c_2, b, p_1 \times \sigma_1(I(r_1))(a), p_2)$.
  end for
  Compute $u_1(\sigma, I(r_1))) = \sum_{a \in A(I(r_1))} \sigma_1(I(r_1))(a) u_1(\sigma, I(r_1), a)$.
  for Each action $a \in A(I(r_1))$ **do**
    $R_1(I, a) = \frac{1}{T+1}\left(TR_1(I, a) + p_2(u_1(\sigma, I(r_1), a) - u_1(\sigma, I(r_1)))\right)$
  end for
  Set $u_1(\sigma, r_1) = u_1(\sigma, I(r_1))$
  Compute $u_2(\sigma, r_2) = \sum_{a \in A(I(r_1))} \sigma_1(I(r_1))(a) u_2(\sigma, r_2, a)$.
else if r_2 is a player node (meaning r_1 is an opponent node) **then**
  do (opposite of above)
else if r_1 is a bucket node **then**
  Choose the child c_1 of r_1 according to the class in b for player 1 on the appropriate round and the child c_2 of r_2 similarly.
  Find $u_1(\sigma, c_1)$ and $u_2(\sigma, c_2)$ from WALKTREES(c_1, c_2, b, p_1, p_2).
  Set $u_1(\sigma, r_1) = u_1(\sigma, c_1)$ and $u_2(\sigma, r_2) = u_2(\sigma, c_2)$.
else if r_1 is a terminal node **then**
  Find $u_1(\sigma, r_1)$ and $u_2(\sigma, r_2)$, the utility of each player if this node is actually reached.
end if

A.5 Poker-Specific Analysis

We first analyze the non-sampling algorithm from Section 3, significantly tightening the presented regret bounds for the specific case of poker games. We then give a regret analysis for the sampling implementation described in Section 4 and used in the experimental results presented in Section 5.

A.5.1 Non-Sampling Algorithm

In Section 3, we discussed Blackwell's Approachability Theorem being applied in every information set. The disadvantage of such an algorithm is that every iteration involves a walk across the entire game tree. The advantage of such an algorithm is that it converges really quickly in terms of iterations. In this analysis, we focus on poker.

If we can bound the difference in any two counterfactual utilities at every information set, we can achieve a bound on the overall regret, because Blackwell's Approachability Theorem gives a guarantee based upon this. In particular, after T time steps, if the bound for the counterfactual utility at an information set is $\Delta_{u,1}(I)$, and there are $|A(I)|$ actions, then the counterfactual regret is bounded by:

$$R_1^T(I) \leq \frac{\Delta_{u,1}(I)\sqrt{|A(I)|}}{\sqrt{T}} \tag{21}$$

By Theorem 3, this means the average overall regret is bounded by:

$$R_1^T \leq \sum_{I \in \mathcal{I}_1} \frac{\Delta_{u,1}(I)\sqrt{|A(I)|}}{\sqrt{T}} \tag{22}$$

First of all, define $\Delta_{u,1}$ to be the overall range of utilities in limit poker (48 small bets/hand). In particular, given $\pi_0(I)$ (the probability of chance acting to reach a node), $\Delta_{u,1}(I) \leq \pi_0(I)\Delta_{u,1}$. In limit one could be more precise, because any information set that begins with both players checking on the pre-flop has a tighter limit on the maximum won or lost, but bounding based on chance nodes is more crucial. In the next step, we leverage the structure of poker: in particular, the fact that all actions are observable. Define $\mathcal{B}_1$ to be the set of all betting sequences where the first player has to act: in particular, $\mathcal{B}_1$ can be considered a partition of the information sets $\mathcal{I}_1$ (such that each $B \in \mathcal{B}_1$ is a set of information sets). Note that, for all $B \in \mathcal{B}_1$:

$$\sum_{I \in B} \pi_0(I) = 1 \tag{23}$$

Moreover, observe that we can define $A(B)$ to be the set of actions available at any information set in B. Applying these concepts to the equation:

$$R_1^T \leq \sum_{B \in \mathcal{B}_1} \frac{\sqrt{|A(B)|}\Delta_{u,1}}{\sqrt{T}} \tag{24}$$

$$R_1^T \leq \frac{\Delta_{u,1}}{\sqrt{T}} \sum_{B \in \mathcal{B}_1} \sqrt{|A(B)|} \tag{25}$$

Thus, *increasing the size of the card abstraction does not affect the rate of convergence*. This is not as surprising as one might think: if one imagined n independent algorithms minimizing regret, each with a bound on their utility of $\Delta_{u,1}$, then one would expect that the theoretical bound on the average of the algorithms would closely resemble the theoretical bound on the average of one particular algorithm. This is very similar to what was leveraged in this section. However, the number of information sets does have an affect on the cost of an iteration: each game state in the abstraction must be traversed in every iteration. This is the primary motivation for WALKTREES.

A.5.2 Sampling Algorithm

In order to analyze WALKTREES, we focus on two different measures of regret:

1. $\hat{R}$, the regret measured by the algorithm.
2. R, the underlying regret (if all states were visited every iteration).

In this implementation, the range of counterfactual utilities can be $\Delta_{u,1}$ in almost every state. Define $C^T(I)$ to be the number of times an information set I was visited until time T: in particular, how many times the bucket sequence that makes I reachable was selected. Blackwell's Approachability Theorem yields us:

$$\hat{R}_1^T(I) \leq \frac{\Delta_{u,1}(I)\sqrt{|A(I)|}\sqrt{C^T(I)}}{T} \tag{26}$$

Note that this is the average regret bound for $C^T(I)$ iterations averaged over T iterations. Observe that for any $B \in \mathcal{B}_1$ (see Section A.5.1), $\sum_{I \in B} C^T(I) = T$. Define $Y = \max_{B \in \mathcal{B}_1} |B|$, in other words the number of card partitions on the river. Then $\sum_{I \in B} \sqrt{C^T(I)} \leq \sqrt{YT}$ (this is because, for arbitrary m, n,

$(a_1 \ldots a_m) \in \mathbf{R}^m$, where $a_i \geq 0$ and $\sum_{i=1}^m a_i = n$, $\sum_{i=1}^m \sqrt{a_i} \leq \sqrt{mn}$).

$$\sum_{I \in B} \hat{R}^T(I) \leq \sum_{I \in B} \frac{\Delta_{u,1}\sqrt{|A(I)|}\sqrt{C^T(I)}}{T} \tag{27}$$

$$\sum_{I \in B} \hat{R}^T(I) \leq \frac{\sqrt{|A(B)|}\Delta_{u,1}\sqrt{Y}}{\sqrt{T}} \hat{R}_1^T \qquad \leq \frac{\Delta_{u,1}\sqrt{Y}}{\sqrt{T}} \sum_{B \in \mathcal{B}_1} \sqrt{|A(B)|} \tag{28}$$

$$\hat{R}_1^T \leq \frac{\Delta_{u,1}\sqrt{Y}}{\sqrt{T}} |\mathcal{B}_1| \sqrt{|A_1|} \tag{29}$$

Thus, the regret bound has increased by a factor of $\sqrt{Y}$: however, the computation per round has decreased by a factor of nearly Y^2, resulting in a dramatic overall gain, so long as R and $\hat{R}$ are similar.

This last portion is tricky: since the algorithm is randomized, we cannot guarantee that every information set is reached, let alone that it has converged. Therefore, instead of proving a bound on the absolute difference of R and $\hat{R}$, we focus on proving a probabilistic connection.

In particular, we will focus on the similarity of the counterfactual regret ($R^T(I)$ and $\hat{R}^T(I)$) in every node. In particular, we will focus on the similarity of the counterfactual regret of a particular action at a particular time ($r_1^t(I,a)$ and $\hat{r}_1^t(I,a)$). Define $\mathrm{Reach}^t(I)$ to be true if I is reachable given the actions of nature at time t. Formally:

$$r_1^t(I,a) = \pi_{-1}^{\sigma^t}(I)\left(u_1(\sigma^t|_{I \to a}, I) - u_1(\sigma^t, I)\right) \tag{30}$$

$$\hat{r}_1^t(I,a) = \begin{cases} \frac{r_1^t(I,a)}{\pi_0(I)} & \text{if } \mathrm{Reach}^t(I) \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

It is the case that $\mathbf{E}[r_1^t(I,a) - \hat{r}_1^t(I,a)] = 0$. These are the elementary components of $R_1^T(I)$ and $\hat{R}^T(I)$, because:

$$R_1^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^T r_1^t(I,a) \tag{32}$$

$$\hat{R}_1^T(I) = \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^T \hat{r}_1^t(I,a) \tag{33}$$

We bound the expected squared difference between $\sum_{I \in \mathcal{I}_i} R_1^T(I)$ and $\sum_{I \in \mathcal{I}_i} \hat{R}_1^T(I)$ in order to prove that they are close, because for any random variable X:

$$\Pr[|X| \geq k\sqrt{\mathbf{E}[X^2]}] \leq \frac{1}{k^2} \tag{34}$$

by Markov's Inequality.

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq |\mathcal{I}_1| \sum_{I \in \mathcal{I}_1} \mathbf{E}[(R_1^T(I) - \hat{R}_1^T(I))^2] \tag{35}$$

This is because, for all $a_1 \ldots a_k \in \mathbf{R}$, $(\sum_{i=1}^k a_i)^2 \leq k \sum_{i=1}^k a_i^2$. Finally:

$$(R_1^T(I) - \hat{R}_1^T(I))^2 = \left(\frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^T r_1^t(I,a) - \frac{1}{T} \max_{a \in A(I)} \sum_{t=1}^T \hat{r}_1^t(I,a)\right)^2 \tag{36}$$

$$(R_1^T(I) - \hat{R}_1^T(I))^2 \leq \frac{1}{T^2} \max_{a \in A(I)} \left(\sum_{t=1}^T r_1^t(I,a) - \sum_{t=1}^T \hat{r}_1^t(I,a)\right)^2 \tag{37}$$

$$(R_1^T(I) - \hat{R}_1^T(I))^2 \leq \frac{1}{T^2} \sum_{a \in A(I)} \left(\sum_{t=1}^T r_1^t(I,a) - \sum_{t=1}^T \hat{r}_1^t(I,a)\right)^2 \tag{38}$$

$$\mathbf{E}[(R_1^T(I) - \hat{R}_1^T(I))^2] \leq \frac{1}{T^2} \sum_{a \in A(I)} \sum_{t=1}^T \mathbf{E}[\left(r_1^t(I,a) - \hat{r}_1^t(I,a)\right)^2] \tag{39}$$

The final step is because if $t \neq t'$, then $\mathbf{E}[\left(r_1^t(I,a) - \hat{r}_1^t(I,a)\right)\left(r_1^{t'}(I,a) - \hat{r}_1^{t'}(I,a)\right)] = 0$. Substituting back into Equation 35:

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq \frac{|\mathcal{I}_1|}{T^2} \sum_{I \in \mathcal{I}_1} \sum_{a \in A(I)} \sum_{t=1}^{T} \mathbf{E}[\left(r_1^t(I,a) - \hat{r}_1^t(I,a)\right)^2] \tag{40}$$

Recall that $\pi_{-1}^{\sigma^t}(I) = \pi_2^{\sigma^t}(I)\pi_0^{\sigma^t}(I)$. Thus, $|r_1^t(I,a)| \leq \Delta_{u,1}\pi_0^{\sigma^t}$, and $\hat{r}_1^t(I,a) \leq \Delta_{u,1}$. Also, $\Pr[\hat{r}_1^t(I,a) \neq 0] \leq \pi_0(I)$. Finally:

$$\mathbf{E}[(r_1^t(I,a) - \hat{r}_1^t(I,a))^2 | \text{Reach}(I)] \leq 2\Delta_{u,1}^2 \tag{41}$$

$$\mathbf{E}[(r_1^t(I,a) - \hat{r}_1^t(I,a))^2 | \neg\text{Reach}(I)] \leq 2\pi_0(I)\Delta_{u,1}^2 \tag{42}$$

$$\mathbf{E}[(r_1^t(I,a) - \hat{r}_1^t(I,a))^2] \leq 4\pi_0(I)\Delta_{u,1}^2 \tag{43}$$

Thus, substituting back into Equation 40:

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq \frac{|\mathcal{I}_1|}{T^2} \sum_{I \in \mathcal{I}_1} \sum_{a \in A(I)} \sum_{t=1}^{T} 4\pi_0(I)\Delta_{u,1}^2 \tag{44}$$

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq \frac{4|\mathcal{I}_1|\Delta_{u,1}^2}{T} \sum_{I \in \mathcal{I}_1} |A(I)|\pi_0(I) \tag{45}$$

$$\tag{46}$$

Again, by focusing on $\mathcal{B}_i$:

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq \frac{4|\mathcal{I}_1|\Delta_{u,1}^2}{T} \sum_{B \in \mathcal{B}_1} \sum_{I \in B} |A(I)|\pi_0(I) \tag{47}$$

$$\mathbf{E}[(\sum_{I \in \mathcal{I}_1} (R_1^T(I) - \hat{R}_1^T(I)))^2] \leq \frac{4|\mathcal{I}_1|\Delta_{u,1}^2}{T} \sum_{B \in \mathcal{B}_1} |A(B)| \tag{48}$$

For any $p \in [0,1]$, with probability at least $1-p$:

$$R_1^T \leq \frac{2\sqrt{|\mathcal{I}_1||\mathcal{B}_1||A_1|}\Delta_{u,1}}{\sqrt{pT}} + \frac{\Delta_{u,1}\sqrt{Y}}{\sqrt{T}}|\mathcal{B}_1|\sqrt{|A_1|} \tag{49}$$