Solving Heads-up Limit Texas Hold'em

Oskari Tammelin,[1] Neil Burch,[2] Michael Johanson[2] and Michael Bowling[2]
[1]http://jeskola.net, ot@iki.fi
[2]Department of Computing Science, University of Alberta
{nburch,johanson,mbowling}@ualberta.ca

Abstract

Cepheus is the first computer program to essentially solve a game of imperfect information that is played competitively by humans. The game it plays is heads-up limit Texas hold'em poker, a game with over 10^{14} information sets, and a challenge problem for artificial intelligence for over 10 years. Cepheus was trained using a new variant of Counterfactual Regret Minimization (CFR), called CFR$^+$, using 4800 CPUs running for 68 days. In this paper we describe in detail the engineering details required to make this computation a reality. We also prove the theoretical soundness of CFR$^+$ and its component algorithm, regret-matching$^+$. We further give a hint towards understanding the success of CFR$^+$ by proving a tracking regret bound for this new regret matching algorithm. We present results showing the role of the algorithmic components and the engineering choices to the success of CFR$^+$.

Introduction

Game theory provides a framework for thinking about decisions in the presence of other agents with applications in security [Tambe, 2011] and robust decision making [Chen and Bowling, 2012]. Extensive-form games describe a large class of such problems where multiple agents must make decisions with imperfect information about the state of the world. A common approach to computing policies in such games is to solve the game by finding a Nash equilibrium: a strategy for each player where each individually maximises their utility against the opponent strategies.

Recently, we announced that heads-up limit Texas hold'em poker (HULHE) is essentially solved [Bowling *et al.*, 2015]. The resulting program Cepheus is the first computer program to solve a non-trivial imperfect information game played competitively by humans. Unlike perfect information games where it is easy to compute an exact solution, imperfect information solutions are usually approximated. Essentially solving a game involves computing an approximation of a Nash equilibrium along with an argument that the approximation is of a sufficient quality relative to the inherent stochasticity in the game. Cepheus is close enough to a Nash equilibrium of the game of HULHE that a lifetime of human play cannot be used to distinguish Cepheus from an exact Nash equilibrium with high confidence.

HULHE has over 10^{14} information sets (i.e., decision points where a player must act). Even after removing strategically identical suit isomorphisms [Billings *et al.*, 2003; Gilpin *et al.*, 2007], the game has over 10^{13} information sets, one thousand times larger than any previously solved imperfect information game. Solving an imperfect information game at this scale poses two challenges: computation and space. Solution techniques generally require space at least on the order of the size of the resulting strategy. For this reason, space-efficient algorithms like Counterfactual Regret Minimization (CFR) [Zinkevich *et al.*, 2007], and its family of algorithms, are a popular choice for large games [Jackson, 2013; Brown *et al.*, 2015]. Even with space efficient CFR algorithms, HULHE would require 262 TiB (assuming 4-byte values) to store the strategy and regret during computation. A second challenge is computation time. The running time of CFR also grows with the size of the game, and so solving HULHE with traditional CFR variants is also infeasible.

Cepheus overcame these limitations using a new algorithm, CFR$^+$ [Bowling *et al.*, 2015; Tammelin, 2014]. While our original work demonstrated the scalability of CFR$^+$ to solve HULHE, it provided no proof of correctness and no theoretical explanation for its improvements. Furthermore, there was little discussion of the engineering choices required to scale to large problems, and no exploration of the technique in other games. In this paper we provide these missing pieces. We prove that CFR$^+$ is sound, with the same asymptotic convergence rate as traditional CFR. However, in practice it results in a drastic reduction in computation time. We explore the nature of this empirical performance improvement on smaller toy games, and prove a tracking regret property on its component algorithm regret-matching$^+$, which hints at a reason for its success. We also describe the details behind our space efficient and massively distributed implementation of CFR$^+$, using streaming compression, which reduces the space requirement for HULHE from 262 TiB to a manageable 10.9 TiB. These algorithmic and engineering advances enabled Cepheus's 900 core-year computation to be distributed on a high-performance cluster using 4800 CPUs and completed in 68 days. The resulting strategy reached an exploitability of 0.986 mbb/g (or less than one thousandth of a big-blind per game).

Background

An extensive-form game is a formal model of the sequential interactions between one or more players. Let P be the set of players. Let H be the set possible game states, represented as the history of actions taken from the initial game state $\varnothing$. We call the state $h \cdot a \in H$ a child of its parent state h. Furthermore, we will say h is an ancestor of descendant state h' or $h \sqsubseteq h'$ when h is a prefix of h'. Let Z be the set of all terminal states. For each non-terminal state h, $A(h)$ gives the set of legal actions, and $p(h) \mapsto P \cup \{c\}$ gives the acting player, where c denotes the "chance player", which represents stochastic events outside of the players' control. $\sigma_c(h, a)$ is the probability that chance will take action $a \in A(h)$ from state h, and is common knowledge. For every $z \in Z$, $u_p(z) \in \Re$ gives the payoff for player p when the game ends in state z. HULHE is a two player zero-sum game, so $P = 1, 2$ and $u_1(z) + u_2(z) = 0$ for all $z \in Z$.

In an imperfect information game, the game specifies an information partition. Let $\mathcal{I}_p$ be a partition of all of the states with player p to act. For any information set $I \in \mathcal{I}_p$, any two states $h, j \in I$ are indistinguishable by player p. Let $I(h)$ be the information set which contains h, let $Z(I, a)$ be the set of all $z \in Z$ such that $h \cdot a \sqsubseteq z$ for some $h \in I$, and let $z[I]$ be the state $h \in I$ such that $h \sqsubseteq z$. HULHE is a game where players have perfect recall, which means that any two states h and j in an information set $I \in \mathcal{I}_p$ have the same sequence of player p information sets and actions. Informally, perfect recall means that a player does not forget their own actions or any information observed before making those actions.

A behaviour strategy $\sigma_p \in \mathbf{\Sigma}_p$ is a function $\sigma_p(I, a) \mapsto \Re$ which defines a probability distribution over valid actions for every information set $I \in \mathcal{I}_p$. For simplicity, we will say $\sigma_p(h, a)$ to mean $\sigma_p(I(h), a)$. A strategy profile $\sigma \in \mathbf{\Sigma}$ is a tuple of strategies, one for each player. Given σ, it is useful to refer to certain products of probabilities. Let $\pi^\sigma(h) = \prod_{j \cdot a \sqsubseteq h} \sigma_{P(j)}(j, a)$ be the joint probability of reaching h if all players follow σ. We use $\pi^\sigma_{-p}(h)$ to refer to the product of terms where $P(h) \neq p$, and $\pi^\sigma_p(h)$ to refer to the product of only the terms where $P(h) = p$. In games with perfect recall, $\pi_p(h) = \pi_p(h')$ for all states h, h' in $I \in \mathcal{I}_p$, so we can also speak of $\pi_p(I)$. We also use $\pi^\sigma(j, h)$ to refer to the product of terms only from j to h.

The expected utility u^σ_p to player p if all players follow σ is $\sum_Z \pi^\sigma(z) u_p(z)$. The expected utility $u^\sigma_p(I, a)$ of taking an action at an information set is $\sum_{z \in Z(I,a)} \pi^\sigma(z) u_p(z)$. We also use an alternative utility function called counterfactual value: $v^\sigma_p(I, a) = \sum_{z \in Z(I,a)} \pi^\sigma_{-p}(z) \pi^\sigma_p(z[I] \cdot a, z) u_p(z)$. Informally, the counterfactual value of information set I and action a for the player acting at I is the expected value assuming the player plays to reach I and choose a.

σ_p is a player p best response to an opponent strategy σ_{-p} if σ_p maximises $u_p^{\langle \sigma_p, \sigma_{-p} \rangle}$. A Nash equilibrium is a strategy profile where all strategies are simultaneously best responses to each other, and an ϵ-Nash equilibrium is a profile where the expected value for each player is within ϵ of the value of a best response strategy. We use the term exploitability to refer to a profile's average loss to a best response across its component strategies. A Nash equilibrium has an exploitability of zero.

Counterfactual Regret Minimization

Consider a repeated decision making problem where we have acted T times in the past by choosing between actions in a set A. Let $\sigma^t(a)$ be our probability of selecting action a at timestep t. Regret is the measurement of how much additional utility might have been gained by following some alternative strategy instead in hindsight. One popular notion, external regret, considers only static actions as the alternative strategies: $R^T(a) = \sum_{t=1}^{T} (v^t(a) - \sum_{b \in A} \sigma^t(b) v^t(b))$. The overall regret is then $R^T = \max_a R^T(a)$.

Regret-matching [Blackwell, 1956] defines a policy $\sigma^t(a) = R^{t-1}(a)^+ / \sum_{b \in A} R^{t-1}(b)^+$ given observed regret prior to the current time, where $x^+ = \max(x, 0)$. An implementation of regret-matching will generally store the regrets $R^t(a)$ for each action, incrementally updating the values using $\Delta R^t(a) = v^t(a) - \sum_{b \in A} \sigma^t(b) v^t(b)$ to compute $R^t(a) = R^{t-1}(a) + \Delta R^t(a)$. Blackwell showed that if the values are drawn from a bounded interval of width L and we use the regret matching policy, we can bound the overall regret $R^T \leq L\sqrt{|A|T}$ for any sequence of values v^t. This implies that the average regret R^T/T approaches zero as T approaches infinity. Such an algorithm can then be used to approximate an equilibrium in two-player zero-sum games, since if both players have at most ϵ average regret, their average strategies form a 2ϵ-Nash equilibrium.

Regret-matching alone is intractable for extensive-form games, since it would require storing regrets for the exponential number of deterministic strategies. CFR [Zinkevich *et al.*, 2007] overcomes this challenge by independently minimizing regret at each information set. It uses regret-matching over the available actions $A(I)$ at an information set I, computing the regrets $R^t(I, a)$ using the counterfactual value $v(I, a)$ of an action a. Its regret with respect to the exponential number of deterministic strategies can be bounded by the sum of the regrets at the linear number of information sets. When employed in self-play, the average strategy of the players are then guaranteed to converge to a Nash equilibrium.

Poker

From the beginning, poker has had an important role in the development of game theory [Borel, 1921; von Neumann, 1928; von Neumann and Morgenstern, 1947; Nash and Shapley, 1950; Kuhn, 1950]. Full-scale poker has been a challenge problem for artificial intelligence, operations research, and psychology, with work going back more than 40 years [Billings *et al.*, 2002]. We examine two variants of poker in this paper: HULHE, which is played by humans, and Rhode Island hold'em, a synthetic game created for research.

HULHE is a two player poker game that consists of four betting rounds, called the pre-flop, flop, turn, and river. Before each game of poker, both players put in some number of chips, called the blinds, into a community pot. One player puts in the small blind, and the other player puts in the big blind, which is twice as many chips as the small blind. In the pre-flop and flop all bets are the size of the big blind, and in the turn and river all bets are twice the size of the big blind. These fixed ratios mean that instead of using chips, we can describe all outcomes in terms of fractions of a big blind.

Each player gets two private cards at the beginning of the game. As the game progresses, public cards are dealt out: three cards on the flop, one card on the turn, and another card on the river. These public cards are also called board cards.

Rhode Island hold'em is a synthetic two-player poker game constructed for artificial intelligence research [Shi and Littman, 2001]. It is a smaller game than HULHE, with three shorter betting rounds, one private player card, and a single public card dealt on each of the last two rounds. Rhode Island hold'em has around 4×10^6 information sets, and was solved in 2005 [Gilpin and Sandholm, 2005].

A New Scalable Approach

Cepheus needed both algorithmic and engineering advances to make the solving of HULHE possible. Each are discussed in-depth below.

The CFR$^+$ Algorithm

Cepheus uses a new variant of CFR, called CFR$^+$, which we will later show converges dramatically faster than CFR. CFR$^+$ involves four changes. First, CFR$^+$ uses a weighted average strategy $\bar{\sigma}_p^T = 2/(T^2 + T)\sum_{t=1}^{T} t\sigma_p^t$, rather than the uniform average used by CFR. Second, many CFR implementations use sampling techniques to speed convergence [Lanctot *et al.*, 2009], whereas CFR$^+$ uses no sampling. Third, CFR as described in the original paper simultaneously updates regret for both players, while CFR$^+$ does alternating updates. Fourth, and most importantly, CFR$^+$ uses regret-matching$^+$ in place of regret-matching.

Regret-matching$^+$ is a regret-minimizing algorithm that operates very similarly to regret-matching. Where regret-matching ignores actions that have an accumulated negative regret, regret-matching$^+$ actively resets any accumulated negative regret back to zero. Formally, regret-matching$^+$ does not store the regrets $R^t(a)$, but instead tracks a regret-like value: $Q^t(a) = \left(Q^{t-1}(a) + \Delta R^t(a)\right)^+$, and bases its policy on these values $\sigma^t(a) = Q^{t-1}(a)/\sum_{b\in A} Q^{t-1}(b)$.

Theorem 1 *Given a set of actions A, and any sequence of T value functions $v^t : A \mapsto \Re$ with a bound L such that $|v^t(a) - v^t(b)| \leq L$ for all t and $a, b \in A$, an agent acting according to the regret-matching$^+$ algorithm will have regret of at most $L\sqrt{|A|T}$.*

The proof of all of the theorems in this paper are in the appendix. So, regret-matching$^+$ has the same regret bound as regret-matching, and therefore CFR$^+$ has the same regret bound as CFR. However, in the next section we show in practice CFR$^+$ dramatically outperforms CFR.

This empirical difference has a partial explanation. Intuitively, one might expect regret-matching$^+$ to outperform regret-matching when the best action suddenly changes. Regret-matching must wait for the previously poor action to prove itself, overcoming all of its accumulated negative regret. Regret-matching$^+$, though, will start playing the new best action immediately since its accumulated negative regret is forgotten. This intuition can be captured by examining the algorithms in terms of tracking regret. Tracking regret [Herbster and Warmuth, 1998] considers the hindsight performance of a larger set of alternatives, e.g., strategies that change their action at most $(k-1)$ times. Regret-matching has very poor tracking regret properties, possibly having linear regret even against strategies with a single switch ($k = 2$). In contrast, regret-matching$^+$ is the first regret-matching based algorithm with sublinear tracking regret.

Theorem 2 *Consider alternative sequences of strategies that can change up to $k-1$ times, then regret-matching$^+$ has a regret bound of $kL\sqrt{|A|T}$.*

The final change in CFR$^+$ is to use a linearly-increasing weighted average strategy.

Theorem 3 *Let σ^t be a sequence of T behaviour strategy profiles from running CFR$^+$ for T iterations, in an extensive-form game where $\max|v_p(l) - v_p(l')| \leq L$ for all players p and terminal histories l, l'. Then the linearly weighted average strategy profile where $\bar{\sigma}_p^T = 2/(T^2+T)\sum_{t=1}^{T} t\sigma_p^t$ is a $2(|\mathcal{I}_1| + |\mathcal{I}_2|)L\sqrt{k}/\sqrt{T}$-Nash equilibrium, where $k = \max_{I\in\mathcal{I}}|A(I)|$.*

This result guarantees that linear weighting CFR$^+$ achieves the same asymptotic guarantees as uniform weighting with CFR or CFR$^+$. Theorem 3 does not apply to CFR, and in contrast to CFR$^+$ where linear weighting improves performance, CFR performance decreases with linear weighting. In the next section we give an experimental comparison of uniform and linear weighting. We also show that in practice the current strategy in CFR$^+$ often achieves a close approximation to the Nash equilibrium, so this aggressive weighting can further speed convergence.

Engineering Details

CFR$^+$ is space efficient, and improves on the speed of CFR, but using CFR$^+$ to solve a game the size of HULHE still poses a challenge both in terms of computation time and space. The 262 TiB of regret values and average strategy is unlikely to fit in RAM, and may not even fit on available disk. 900 core-years of computation time [Bowling *et al.*, 2015] is too long for a single machine. The approach used by Cepheus is to compress and store the values on disk, and distribute the CFR$^+$ computation across a cluster of compute nodes. Both parts of this approach have a number of important technical considerations that we discuss below.

Compressed Values on Disk Because CFR$^+$ updates all values in the game at each iteration, rather than updating randomly selected locations, CFR$^+$ is particularly suited for use with compression. The algorithm amortizes the compression/decompression overhead by doing a full and exact CFR update. For the same reason, CFR$^+$ can efficiently load and save this compressed data to disk. Taking advantage of this to store compressed values on disk still requires attention to a few implementation details. Any compression method used to reduce the storage requirements must simultaneously be good enough to reduce the data to a manageable size, and fast enough that the compute nodes are not excessively waiting on loading and storing the values. In a similar vein, it is critical that disk I/O be serialized and data pre-fetched to minimize the CPUs waiting on the disk. Finally, because space is tight, some care must be taken in managing compressed, uncompressed, and newly compressed data.

We suggest two application specific ideas for combining CFR$^+$ and compression. The entropy of the original data in CFR$^+$ can be reduced, and it is possible to take advantage of game-specific regularity in the data that standard compression techniques may be otherwise unable to find. Both ideas were mentioned in our work on solving HULHE [Bowling *et al.*, 2015], but here we give more detail on these critical enhancements.

First, the entropy of the regrets and average strategy can be reduced by storing values as fixed point numbers instead of floating point numbers. Floating point numbers are an obvious choice for a large computation involving expected values across probabilistic outcomes, but storing the floating point values contain more information than we need for the targeted solution quality. We can instead use floating point numbers for all intermediate computation, but scale the values and truncate them to integers before storing them. By adjusting the scaling factor, we can trade potentially decreased accuracy for fewer significant bits and more easily compressible data. In our results section, we will further explore the effects of the scaling factor.

Games may have some regular structure that a generic compressor has difficulty finding. HULHE has many groups of cards that are likely to be similar. Given a fixed betting sequence and board cards, there are private cards where we would expect to see similar regret values and action probabilities. For some board cards, we would expect values to be similar for all private cards. By sorting the boards, and then sorting the private cards, we can use earlier values to predict the next value. Instead of compressing the values directly, we can compress the errors in the predicted values. The error values take advantage of game specific knowledge encoded in the sorting order, and are more readily compressible. For HULHE, we suggest sorting the boards by the number of suits that can make a flush in each round, breaking ties by card rank, and sorting the private cards by rollout hand strength – the expected number of times the private cards are ahead given all possible opponent cards and future board cards.

With compressed data, it becomes necessary to decompress the data to run the computation, and then re-compress new results. There are two memory management issues that need to be handled carefully. First, decompressing all of the data may require too much space. This can be handled by using just-in-time decompression and compression, decompressing only the necessary blocks of values before re-compressing the data. Second, using separate space for old and newly compressed data doubles the required space. This can be handled by using a linked list of fixed size chunks to store compressed data, recycling chunks from old compressed data after they have been completely consumed by the decompressor.

Distributed Computation CFR$^+$ is a large recursive computation: the counterfactual values used to update regrets at an information set I are combined with the current policy at I to produce one of the counterfactual values for the parent of I. Because of this, the game can be split into a trunk and subgames located on separate machines [Johanson *et al.*, 2011], with probabilities coming in from the trunk, and counterfactual values coming back from the subgames.

If we split the game of HULHE after the second round of betting there are 63 betting sequences and 1755 possible combinations of board cards, for a total of $110,565$ subgames. There are at most $2 * \binom{49}{2} = 2352$ probabilities to pass to each subgame, one for each possible combination of private cards for each player. Each subgame passes the same number of counterfactual values back to the trunk. This is only 1 GiB of communication in each direction per iteration, even using 8-byte floating point values. With full unsampled CFR$^+$ iterations, the cost for 2 GiB of network communication is negligible.

The architecture for Cepheus has one node processing the trunk, sending probabilities for subgames to worker nodes, and then waiting on subgame values from the worker nodes. Each worker node has one thread receiving subgame probabilities from the trunk node and reading the necessary compressed subgames off disk to fill a buffer of subgames to process. Worker nodes have another thread which empties a buffer of processed subgames by writing them to disk. Finally, worker nodes have multiple threads which remove subgames from the common pool of unprocessed subgames, do the CFR$^+$ update, send values to the trunk node, and add the updated subgame to the buffer of processed subgames.

Empirical Results

In this section, our experiments will demonstrate the empirical advantages of CFR$^+$ over CFR, and highlight the engineering decisions that must be made for large-scale implementations. Our domain for these experiments is Rhode Island hold'em, a small synthetic poker game with a similar structure to HULHE. We also demonstrate the strong performance of CFR$^+$ is not limited to poker-like games by comparing CFR and CFR$^+$ in matrix games.

We begin our empirical analysis by considering the two key differences between CFR$^+$ and CFR: using regret-matching$^+$instead of regret-matching, and applying a linear weight to the average strategy updates instead of a constant weight. In Figure 1a, we show the convergence of the average strategies generated by CFR$^+$ and CFR, as well as variants that make only one of these changes. This result illustrates that applying regret-matching$^+$alone provides an advantage over CFR in this game, and also applying the linear weight gives a further improvement.

In Figure 1b, we compare the convergence of the CFR and CFR$^+$ average strategies against the exploitability of their current strategies. The theoretical proofs for CFR and CFR$^+$ guarantee that the average strategy converges towards a Nash equilibrium. While the CFR current strategy does not converge to a Nash equilibrium in theory or in practice, in this figure we observe that the CFR$^+$ current strategy appears to

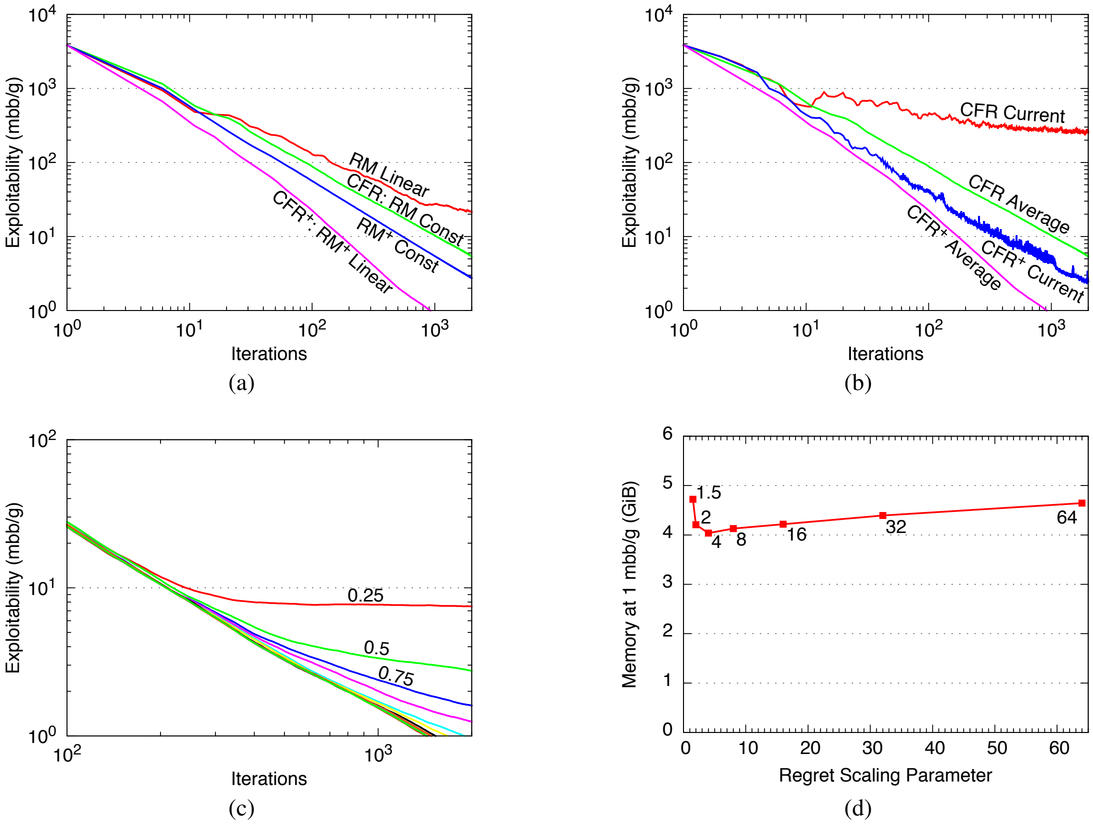


Figure 1: Empirical results and engineering choices in Rhode Island hold'em: (a) convergence rate of CFR using regret-matching (RM) and regret-matching$^+$ (RM$^+$) and linear and constant weighting; (b) convergence rate of the average and current strategies with CFR and CFR$^+$; (c) convergence rate of CFR$^+$ with different scaling factors; and (d) amount of memory used by CFR$^+$ with different scaling factors.

converge in practice in this game, and at a faster rate than the CFR average strategy. Cepheus exploited this observation in solving HULHE, in which the CFR$^+$ current strategy was used as its solution, ultimately converging more quickly than the average.

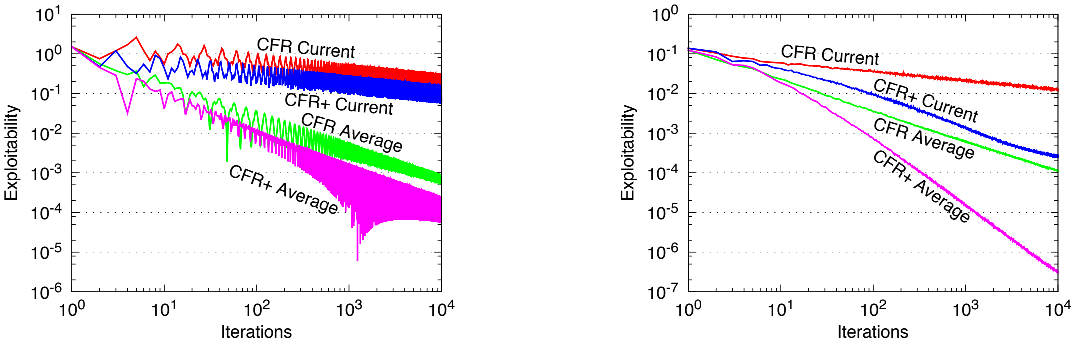
Next, we explore the use of scaling parameters for fixed-point arithmetic involving regret values in our CFR$^+$ implementation. Lower regret scaling factors result in less accurate values, but also enables better compression so that less space is used. For the HULHE solution, it was necessary to carefully tune this parameter so that the computation would reach the target exploitability of 1 milli-big-blind per game while remaining within available storage. In Figure 1c, we demonstrate the first part of this tradeoff in Rhode Island hold'em. Each curve shows the convergence of the CFR$^+$ average strategy when the regret values use a scaling parameter from the set: $\{0.25, 0.5, 0.75, 1, 1.5, 2, 4, 8, 16, 32, 64\}$. Note that the chart focuses on iterations 100 through 2000; before iteration 100, all of the curves followed the same trajectory. However, after iteration 100, we see the curves with low parameter values diverge in increasing order, with four unable to reach the goal of 1 milli-big-blind per game within 2000 iterations.

In Figure 1d, we consider the memory required for each of the parameter choices, at the iteration when they reached an exploitability of 1 milli-big-blind per game. While conservative (and therefore large) scaling factors converge reliably in Figure 1c, they also require more memory than smaller parameter choices. However, overly aggressively small parameters may also require more memory if they converge more slowly and must store larger regret values accumulated over additional iterations. Through experiments in small games, this parameter can be tuned to make a large-scale computation feasible as was done by Cepheus.

Matrix Games

The strong performance of CFR$^+$ is not limited to a few variants of poker. We use matrix games to examine the wider applicability of CFR$^+$ and regret-matching$^+$.

Figure 2a shows the performance of CFR and CFR$^+$ in the matching pennies game, a two by two matrix game. If HULHE is at one extreme as a large, complex games, two by two matrix games are at the other extreme. In the matching pennies game, each player has a coin with two sides, heads and tails. Both players secretly place their coin with either heads or tails facing up. After the coins are placed, both players reveal their coin. We used skewed zero-sum payoffs, where the row player wins 1 if both coins were heads, wins 4 if both coins were tails, and otherwise loses 2.



(a)

(b)

Figure 2: Convergence rate of CFR and CFR$^+$ in: (a) the matching pennies game (b) 1000 by 1000 random matrix games.

While the benefit of CFR$^+$ shown in Figure 2a is not as large as in HULHE or Rhode Island hold'em, CFR$^+$ is still producing low exploitability strategies more quickly than CFR. For example, given a target exploitability of 10^{-3}, CFR first reaches the target by iteration $3,539$, while CFR$^+$ reaches the same target by iteration 343. The very different nature of the matching pennies game compared to HULHE also shows up in the large fluctuations in the exploitability of the average strategy, compared to the visibly smooth progress in larger games.

Figure 2b shows results from a game somewhere between HULHE and the matching pennies game, using 10 randomly generated, 1000 by 1000 matrix games. Payoffs for each pair of strategies in a game were identically and independently sampled from a normal distribution with mean 0 and a standard deviation of 1. We ran CFR and CFR$^+$ on each of the 10 games, and the exploitabilities in Figure 2b are the average values across the 10 runs.

The large performance advantage of CFR$^+$ is visible again in larger matrix games. A target exploitability of 10^{-3} is achieved by CFR in 510 iterations and by CFR$^+$ in 83 iterations. CFR does not reach a target of 10^{-4} at all within $10,000$ iterations, while CFR$^+$ requires only 331 iterations to reach 10^{-4}, and reaches 10^{-6} by iteration $4,836$.

Conclusion

Good play in imperfect information games like poker requires deception and bluffing, which are not generally considered to be machine-like traits. HULHE is the first competitively played game of imperfect information to be essentially solved, marking a significant milestone in game theory and artificial intelligence. The computation behind this result posed both algorithmic and engineering challenges. Cepheus solved HULHE using a new algorithm CFR$^+$, fast custom compression to reduce space, careful use of disk to increase available space, and 4800 CPUs on a high performance research cluster. We demonstrate that the CFR$^+$ algorithm outperforms the commonly used CFR algorithm, prove that CFR$^+$ and its component algorithm regret-matching$^+$ are sound, and show that regret-matching$^+$minimizes a regret measure where traditional regret-matching does not. This establishes the new state-of-the-art in solving large extensive-form games.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council (NSERC), Alberta Innovates Centre for Machine Learning (AICML), and Alberta Innovates Technology Futures (AITF). Computing resources were provided by Calcul Québec, Westgrid, and Compute Canada.

Proofs

Lemma 1 *Given a sequence of strategies $\sigma^1, ..., \sigma^T$, each defining a probability distribution over a set of actions A, let $Q^t(a) = \left(Q^{t-1}(a) + \Delta R^t(a)\right)^+$ and $Q^0(a) = 0$ for all actions $a \in A$. The regret-like value $Q^t(a)$ is then an upper bound on the regret $R^t(a)$, and $Q^t(a) - Q^{t-1}(a) \geq \Delta R^t(a) = R^t(a) - R^{t-1}(a)$.*

Proof For any $t \geq 1$ we have

$$\begin{aligned} Q^{t+1}(a) - Q^t(a) &= \max\left(Q^t(a) + \Delta R^{t+1}(a), 0\right) - Q^t(a) \\ &\geq Q^t(a) + \Delta R^{t+1}(a) - Q^t(a) = R^{t+1}(a) - R^t(a) \end{aligned}$$

This gives us $Q^t(a) = \sum_{i=1}^t Q^i(a) - Q^{i-1}(a) \geq \sum_{i=1}^t R^i(a) - R^{i-1}(a) = R^t(a)$ □

Lemma 2 *Given a set of actions A, and any sequence of T value functions $v^t : A \mapsto \Re$ with a bound L such that $|v^t(a) - v^t(b)| \leq L$ for all t and $a, b \in A$, after playing the sequence σ^t of regret-matching$^+$ strategies, the regret-like value $Q^T(a) \leq L\sqrt{|A|T}$ for all $a \in A$.*

Proof

$$\begin{aligned} (\max_a Q^T(a))^2 = \max_a Q^T(a)^2 &\leq \sum_a Q^T(a)^2 \\ = \sum_a ((Q^{T-1}(a) + v^T(a) - \sum_b \sigma^T(b)v^T(b))^+)^2 \\ \leq \sum_a (Q^{T-1}(a) + v^T(a) - \sum_b \sigma^T(b)v^T(b))^2 \end{aligned}$$

$$= \sum_a (Q^{T-1}(a)^2 + (v^T(a) - \sum_b \sigma^T(b)v^T(b))^2$$
$$+2Q^{T-1}(a)(v^T(a) - \sum_b \sigma^T(b)v^T(b)))$$
$$\le \sum_a Q^{T-1}(a)^2 + |A|L^2$$
$$+2(\sum_a Q^{T-1}(a)v^T(a) - \sum_{a,b} Q^{T-1}(a)v^T(b)\frac{Q^{T-1}(b)}{\sum_c Q^{T-1}(c)})$$
$$= \sum_a Q^{T-1}(a)^2 + |A|L^2$$
$$+2(\sum_a Q^{T-1}(a)v^T(a) - \sum_b v^T(b)Q^{T-1}(b)\sum_c \sigma^T(c))$$
$$= \sum_a Q^{T-1}(a)^2 + |A|L^2$$

$Q^0(a) = 0$ for all a, so by induction $(\max_a Q^T(a))^2 \le T|A|L^2$, which gives us $Q^T(a) \le L\sqrt{|A|T}$. □

Proof of Theorem 1 From Lemma 2, we have $Q^T(a) \le L\sqrt{|A|T}$. From Lemma 1, we get $R^T(a) \le L\sqrt{|A|T}$. This holds for all $a \in A$, so regret $R^T \le L\sqrt{|A|T}$. □

Proof of Theorem 2 Consider an arbitrary k-partition strategy $\vec{s} = s^1, ..., s^T$ for T time steps, with $s^i \in A$, that switches actions at most $k-1$ times. If we played policies $\sigma^1, ..., \sigma^T$ on the T time steps, the regret for $\vec{s}$ is $R^T_{\vec{s}} = \sum_{t=1}^T v^t(s^t) - \sum_{a\in A} \sigma^t(a)v^t(a)$.

For this arbitrary k-partition strategy, we can construct a partition $\mathcal{B}$ of the T time steps into contiguous blocks of time such that for any $B \in \mathcal{B}$ we have $|\mathcal{B}| \le k$ and$s^i = s^j$ for any $i, j \in B$. Let $f[B]$ and $l[B]$ be the first and last time step of any block $B \in \mathcal{B}$, and s^B be the action chosen by $\vec{s}$ at all time steps in block B. We can re-write the regret for $\vec{s}$ as $R^T_{\vec{s}} = \sum_{B\in\mathcal{B}} \sum_{t=f[B]}^{l[B]} \Delta R^t(s^B)$.

Consider the quantity $R^B(s^B) = \sum_{t=f[B]}^{l[B]} \Delta R^t(s^B)$. From Lemma 1 $R^B(s^B) \le \sum_{t=f[B]}^{l[B]} \Delta Q^t(s^B)$. By the definition of regret-matching$^+$ we get $R^B(s^B) \le \sum_{t=1}^{l[B]} \Delta Q^t(s^B) = Q^{l[B]}$. From Lemma 2, $R^B(s^B) \le L\sqrt{|A|l[B]} \le L\sqrt{|A|T}$.

Because we have a bound on $R^B(s^B)$ that does not depend on B, we have $R^T_{\vec{s}} \le |\mathcal{B}|L\sqrt{|A|T} \le kL\sqrt{|A|T}$. Finally, $\vec{s}$ was an arbitrary k-partition strategy, so this bound holds for all k-partition strategies, including the strategy with maximum regret. □

Lemma 3 *Call a sequence $x_1, ..., x_T$ of bounded real values B-plausible if $B > 0$, $\sum_{t=1}^i x_t \ge 0$ for all i, and $\sum_{t=1}^T x_t \le B$. For any B-plausible sequence, $\sum_{t=1}^T tx_t \le TB$.*

Proof Consider any B-plausible sequence that maximises the weighted sum. That is, let $x^*_1, ..., x^*_T = \text{argmax}_{x'_1,...,x'^T} \sum_{t=1}^T tx'_t$. We will show this by proving that $x^*_i \ge 0$ for all $1 \le i \le T$.

For any $i < T$, it can not be the case that for any $i < j$ that $x^*_i > 0$, $x^*_j < 0$, and $x^*_k \ge 0$ for all k where $i < k < j$. Assume this were true, then let $\delta = \min(|x^*_i|, |x^*_j|)$. Construct a new sequence x' where $x'_i = x^*_i - \delta$, $x'_j = x^*_j + \delta$, and $x'_k = x^*_k$ for all $k \ne i, j$. This new sequence x' is B-plausible. For all k where $i \le k < j$ we have $\sum_{t=1}^k x'_t = -\delta + \sum_{t=1}^{i-1} x^*_t + \sum_{t=i}^k x^*_t \ge -\delta + 0 + \delta = 0$. For all k where $k < i$ or $k \ge j$ we have $\sum_{t=1}^k x'_t = \sum_{t=1}^k x^*_t$. Looking at the weighted sum of x', we also have $\sum_t tx'_t \ge \sum_t tx^*_t$, which contradicts the construction of x^* as a maximizing sequence.

Further, it can not be the case that $x^*_j < 0$ for any j. Assume there is a negative value. Let j be the minimum index such that that $x^*_j < 0$. Because j is the minimum index, $x^*_k \ge 0$ for all $k < j$. From above, it can not be the case that $x^*_i > 0$ for any $i < j$. This means $x^*_k = 0$ for all $k < j$, so we have $\sum_{t=1}^j x^*_t = x^*_j < 0$, which contradicts x^* being a B-plausible sequence. Therefore, we have $x^*_i \ge 0$ for all i.

Using the fact all values in the sequence are non-negative, $\sum_t tx^*_t \le \sum_t Tx^*_t = T\sum_t x^*_t$. From the definition of B-plausible, $\sum_t x^*_t \le B$, so $\sum_t tx^*_t \le TB$. □

Lemma 4 *Let A be a set of actions, $v^t : A \mapsto \Re$ be a sequence of T value functions over A with a bound L such that $|v^t(a) - v^t(b)| \le L$ for all t and $a, b \in A$, and σ^t be the sequence of regret-matching$^+$ policies over T time steps. Construct a new sequence v'^t of $(T^2+T)/2$ value functions by using t subsequent copies of v^t (i.e., $v^1, v^2, v^2, v^3, v^3, v^3, ...$) and a similar sequence σ'^t from σ^t. Then the regret $R^T(a)$ of sequence σ' is bounded by $TL\sqrt{|A|T}$.*

Proof For any action a, consider the regret $R'^{(T^2+T)/2}(a)$ of the expanded sequences v' and σ'. We know $R'^{(T^2+T)/2}(a) = \sum_{t=1}^{(T^2+T)/2} \Delta R'^t(a)$ where $\Delta R'^t(a) = v'^t(a) - \sum_{b\in A} \sigma'^t(b)v'^t(b)$. By construction of v' and σ', the sequence $\Delta R'^t(a)$ is a sequence of t subsequent copies of $\Delta R^t(a) = v^t(a) - \sum_{b\in A} \sigma^t(b)v^t(b)$. So $R'^{(T^2+T)/2}(a) = \sum_{t=1}^T t\Delta R^t(a)$.

Let $\Delta Q^t(a) = Q^t(a) - Q^{t-1}(a)$ be the change in Q values for action a from using regret-matching$^+$ on the T time steps with value functions v^t. From Lemma 1, we have $\Delta Q^t(a) \ge \Delta R^t(a)$. From above, this gives us $R'^{(T^2+T)/2}(a) \le \sum_{t=1}^T t\Delta Q^t(a)$. $\Delta Q^t(a)$ is a $L\sqrt{|A|T}$-plausible sequence, so by Lemma 3 $R'^{(T^2+T)/2}(a) \le \sum_{t=1}^T t\Delta Q^t(a) \le TL\sqrt{|A|T}$. □

Proof of Theorem 3 From the sequence of T strategy profiles σ^t, construct a new sequence σ'^t of $(T^2+T)/2$ strategy profiles consisting of t subsequent copies of σ^t, as in Lemma 3. From Lemma 4, we know that for any information set I and action a, $R'^{(T^2+T)/2}(I, a) \le TL\sqrt{|A(I)|T}$. Because this holds for arbitrary a, we have $R'^{(T^2+T)/2}(I) \le TL\sqrt{|A(I)|T}$.

From the original CFR convergence proof [Zinkevich *et al.*, 2007], we have $R_p'^{(T^2+T)/2} \leq \sum_{I\in\mathcal{I}_p} R'^{(T^2+T)/2}(I)$ for any player p. From above, $R_p'^{(T^2+T)/2} \leq \sum_{I\in\mathcal{I}_p} TL\sqrt{|A(I)|T} \leq |\mathcal{I}_p|TL\sqrt{kT}$. Because $T^2+T \geq T^2$, we get the average regret $2R_p'^{(T^2+T)/2}/(T^2+T) \leq 2|\mathcal{I}_p|L\sqrt{k}/\sqrt{T}$.

From the folk theorem linking regret to an ϵ-Nash equilibrium, given a sequence of strategy profiles with player 1 and 2 average regrets of a and b, the strategy profile of the average strategies is a $(a+b)$-Nash equilibrium. The average strategies from the sequence σ' is therefore a $2(|\mathcal{I}_1|+|\mathcal{I}_2|)L\sqrt{k}/\sqrt{T}$-Nash equilibrium.

Finally, by the construction of σ', for any player p the average strategy $2/(T^2+T)\sum_1^{(T^2+T)/2}\sigma_p'^t = 2/(T^2+T)\sum_1^T t\sigma_p^t = \bar{\sigma}_p^T$, which is the linearly weighted average strategy defined in statement of the theorem. □

References

[Billings *et al.*, 2002] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134(1–2):201–240, 2002.

[Billings *et al.*, 2003] Darse Billings, Neil Burch, Aaaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 661–668, 2003.

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

[Borel, 1921] Émile Borel. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes Rendus de l'Académie des Sciences*, 173:1304–1308, 1921.

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

[Brown *et al.*, 2015] 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 *International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[Chen and Bowling, 2012] Katherine Chen and Michael Bowling. Tractable objectives for robust policy optimization. In *Advances in Neural Information Processing Systems 25 (NIPS)*, pages 2078–2086, 2012.

[Gilpin and Sandholm, 2005] Andrew Gilpin and Tuomas Sandholm. Optimal Rhode Island hold'em poker. In *Proceedings of the Twentieth AAAI Conference on Artificial Intelligence*, pages 1684–1685, 2005.

[Gilpin *et al.*, 2007] Andrew Gilpin, Tuomas Sandholm, and Troels Bjerre Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of Texas hold'em poker. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 50–57, 2007.

[Herbster and Warmuth, 1998] Mark Herbster and Manfred K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, August 1998.

[Jackson, 2013] Eric Griffin Jackson. Slumbot NL: Solving large games with counterfactual regret minimization using sampling and distributed processing. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[Johanson *et al.*, 2011] 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*, pages 258–265, 2011.

[Kuhn, 1950] H.W. Kuhn. Simplified two-person poker. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games*, volume 1 of *Annals of mathematics studies*, pages 97–103. Princeton University Press, 1950.

[Lanctot *et al.*, 2009] 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*, pages 1141–1149, 2009.

[Nash and Shapley, 1950] J. F. Nash and L. S. Shapley. A simple 3-person poker game. In *Contributions to the Theory of Games I*, pages 105–116. Princeton University Press, 1950.

[Shi and Littman, 2001] Jiefu Shi and Michael Littman. Abstraction methods for game theoretic poker. In *Computers and Games*, pages 333–345. Springer, 2001.

[Tambe, 2011] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[Tammelin, 2014] Oskari Tammelin. CFR+. *CoRR*, abs/1407.5042, 2014.

[von Neumann and Morgenstern, 1947] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 1947.

[von Neumann, 1928] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[Zinkevich *et al.*, 2007] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, pages 905–912, 2007.