MCRNR: Fast Computing of Restricted Nash Responses by Means of Sampling

Marc Ponsen[1] and **Marc Lanctot**[2] and **Steven de Jong**[13]
1: Department of Knowledge Engineering, Maastricht University, Maastricht, Netherlands
2: Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada
3: Computational Modeling Lab, Vrije Universiteit Brussel, Brussel, Belgium

Abstract

This paper presents a sample-based algorithm for the computation of restricted Nash strategies in complex extensive form games. Recent work indicates that regret-minimization algorithms using selective sampling, such as Monte-Carlo Counterfactual Regret Minimization (MCCFR), converge faster to Nash equilibrium (NE) strategies than their non-sampled counterparts which perform a full tree traversal. In this paper, we show that MCCFR is also able to establish NE strategies in the complex domain of Poker. Although such strategies are defensive (i.e. safe to play), they are oblivious to opponent mistakes. We can thus achieve better performance by using (an estimation of) opponent strategies. The Restricted Nash Response (RNR) algorithm was proposed to learn robust counter-strategies given such knowledge. It solves a modified game, wherein it is assumed that opponents play according to a fixed strategy with a certain probability, or to a regret-minimizing strategy otherwise. We improve the rate of convergence of the RNR algorithm using sampling. Our new algorithm, MCRNR, samples only relevant parts of the game tree. It is therefore able to converge faster to robust best-response strategies than RNR. We evaluate our algorithm on a variety of imperfect information games that are small enough to solve yet large enough to be strategically interesting, as well as a large game, Texas Hold'em Poker.

Introduction

This paper presents MCRNR, a sample-based algorithm for the computation of restricted Nash strategies in complex extensive form games. This algorithm combines the advantages of two state-of-the-art existing algorithms, i.e., (Monte-Carlo) Counterfactual Regret Minimization (CFR) (Zinkevich et al. 2008) (Lanctot et al. 2009) and Restricted Nash (RNR) (Johanson, Zinkevich, and Bowling 2008) (Johanson and Bowling 2009).

CFR and MCCFR. The CFR algorithm was developed to find approximate NE strategies in complex games (Zinkevich et al. 2008). Most notably, using CFR, NE strategies were computed in an abstracted version of the highly complex game of 2-player limit Texas Hold'em Poker. The algorithm traverses the whole game tree, and updates the strategy at each iteration. For larger games doing a full tree traversal becomes impractical due to time and memory constraints. Researchers therefore investigated whether NE strategies can be computed by means of sampling in the tree. Indeed, the sampling-based algorithm Monte-Carlo Counterfactual Regret Minimization (MCCFR) was shown to converge faster to NE strategies than algorithms that perform a full tree traversal (Lanctot et al. 2009). In this paper, we show that MCCFR also finds NE strategies in very large games, by applying it to 2-player limit Texas Hold'em.

RNR. Unlike NE strategies that are oblivious to opponent play, RNR strategies are robust best-response strategies given a model of opponent policies (Johanson, Zinkevich, and Bowling 2008). It was shown that RNR strategies are capable of exploiting opponents, with reasonable performance even when the model is wrong. The RNR algorithm solves a modified game, wherein it is assumed that the opponent plays according to a fixed strategy, as specified by the model, with a certain probability. Otherwise, the opponent plays according to a regret-minimizing strategy.[1]

Our contribution: MCRNR. Given the promising results of applying sampling in CFR, we extend the original RNR algorithm with sampling. Our new algorithm, MCRNR, benefits from sampling only relevant parts of the game tree. The algorithm is therefore able to converge very quickly to robust best-response strategies in example Poker-like games, as well as in the large complex game of 2-player limit Texas Hold'em Poker.

Background

In this section we give an overview of the concepts required to describe our algorithm. For details, refer to (Osborne and Rubinstein 1994). An extensive game is a general model of sequential decision-making with imperfect information. As with perfect information games (such as Chess or Checkers), extensive games consist primarily of a game tree: each non-terminal node has an associated player (possibly chance) that makes the decision at that node, and each terminal node

[1] The Data-Biased Response algorithm is similar, except that the choice of whether the opponent plays according to the model is done separately at each information set. The confidence of the model depends on the visiting frequencies of information sets (Johanson and Bowling 2009).

has associated utilities for the players. Additionally, game states are partitioned into information sets $\mathcal{I}_i$ where a player i cannot distinguish between states in the same information set. The players, therefore, must choose actions with the same distribution at each state in the same information set. In this paper, we focus on 2-player extensive games.

A **strategy** of player i, σ_i, is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathcal{I}_i$, where I_i is an information set belonging to i, and $A(I_i)$ is the set of actions that can be chosen at that information set. We denote Σ_i as the set of all strategies for player i. A **strategy profile**, σ, consists of a strategy for each player, $\sigma_1, \ldots, \sigma_n$. We let σ_{-i} refer to the strategies in σ excluding σ_i.

Valid sequences of actions in the game are called **histories**, denoted $h \in H$. A history is a **terminal history**, $h \in Z$ where $Z \subset H$, if its sequences of actions lead from root to leaf. A **prefix history** $h \sqsubseteq h'$ is one where h' can be obtained by taking a valid sequence of actions from h. Given h, the current player to act is denoted $P(h)$. Each information set contains one or more valid histories. We assume the standard assumption of perfect recall: information sets are defined by the information that was revealed to each player over the course of a history assuming infallible memory.[2]

Let $\pi^\sigma(h)$ be the probability of history h occurring if all players choose actions according to σ. We can decompose $\pi^\sigma(h)$ into each player's contribution to this probability. Here, $\pi_i^\sigma(h)$ is the contribution to this probability from player i when playing according to σ. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except that of player i. Finally, let $\pi^\sigma(h,z) = \pi^\sigma(z)/\pi^\sigma(h)$ if $h \sqsubseteq z$, and zero otherwise. Let $\pi_i^\sigma(h,z)$ and $\pi_{-i}^\sigma(h,z)$ be defined similarly. Using this notation, we can define the expected payoff for player i as $u_i(\sigma) = \sum_{h \in Z} u_i(h)\pi^\sigma(h)$.

Given a strategy profile, σ, we define a player's **best response** as a strategy that maximizes their expected payoff assuming all other players play according to σ. The **best-response value** for player i is the value of that strategy, $b_i(\sigma_{-i}) = \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i})$. An **$\epsilon$-Nash equilibrium** is an approximation of a Nash equilibrium; it is a strategy profile σ that satisfies

$$\forall i \in N \quad u_i(\sigma) + \epsilon \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{-i}) \tag{1}$$

If $\epsilon = 0$ then σ is a **Nash equilibrium**: no player has any incentive to deviate as they are all playing best responses. If a game is two-player and zero-sum, we can use **exploitability** as a metric for determining how close σ is to an equilibrium, $\epsilon_\sigma = b_1(\sigma_2) + b_2(\sigma_1)$.

Monte Carlo CFR

Imagine the situation where players are playing with strategy profile σ. Players may regret using their strategy σ_i against σ_{-i} to some extent. In particular, for some information set I they may regret *not* taking a particular action a instead of following σ_i. Let $\sigma_{I \to a}$ be a strategy identical to σ except a is taken at I. Let Z_I be the subset of all terminal histories where a prefix of the history is in the set I; for $z \in Z_I$ let $z[I]$ be that prefix. Since we are restricting ourselves to perfect recall games $z[I]$ is unique.[2] The counterfactual value $v_i(\sigma, I)$ is defined as:

$$v_i(\sigma, I) = \sum_{z \in Z_I} \pi_{-i}^\sigma(z[I])\pi^\sigma(z[I], z)u_i(z). \tag{2}$$

Counterfactual regret minimization (CFR) algorithm applies a no-regret learning policy at each information set over these counterfactual values (Zinkevich et al. 2008). Each player starts with an initial strategy and accumulates a counterfactual regret for each action at each information set $r(I,a) = v(\sigma_{I \to a}, I) - v(\sigma, I)$ through self-play (game tree traversal). Minimizing the regret of playing σ_i at each information set also minimizes the overall external regret, and so the average strategies approach a Nash equilibrium.

Monte-Carlo Counterfactual Regret Minimization (MCCFR) (Lanctot et al. 2009) avoids traversing the entire game tree on each iteration while still having the immediate counterfactual regrets be unchanged *in expectation*. Let $\mathcal{Q} = \{Q_1, \ldots, Q_r\}$ be a set of subsets of Z, such that their union spans the set Z. These Q_j are referred to as blocks of terminal histories. MCCFR samples one of these blocks and only considers the terminal histories in the sampled block. Let $q_j > 0$ be the probability of considering block Q_j for the current iteration (where $\sum_{j=1}^{r} q_j = 1$).

Let $q(z) = \sum_{j:z \in Q_j} q_j$, i.e., $q(z)$ is the probability of considering terminal history z on the current iteration. The sampled counterfactual value when updating block j is:

$$\tilde{v}_i(\sigma, I|j) = \sum_{z \in Q_j \cap Z_I} \frac{1}{q(z)} \pi_{-i}^\sigma(z[I])\pi^\sigma(z[I], z)u_i(z) \tag{3}$$

Selecting a set $\mathcal{Q}$ along with the sampling probabilities defines a complete sample-based CFR algorithm. Rather than doing full game tree traversals the algorithm samples one of these blocks, and then examines only the terminal histories in that block.

Sampled counterfactual matches counterfactual value on expectation. That is, $\mathbb{E}_{j \sim q_j}[\tilde{v}_i(\sigma, I|j)] = v_i(\sigma, I)$. So, MCCFR samples a block and for each information set that contains a prefix of a terminal history in the block we compute the *sampled immediate counterfactual regrets* of each action, $\tilde{r}(I,a) = \tilde{v}_i(\sigma^t_{(I \to a)}, I) - \tilde{v}_i(\sigma^t, I)$. These sampled counterfactual regrets are accumulated, and the player's strategy on the next iteration is determined by the regret-matching rule to the accumulated regrets (Hart and Mas-Colell 2000). This rule assigns a probability to an action in an information set. Define $r_I^+[a] = \max\{0, r_I[a]\}$. Then:

$$\sigma(I,a) = \begin{cases} 1/|A(I)| & \text{if } \forall a \in A(I): r_I[a] \leq 0 \\ \frac{r_I^+[a]}{\sum_{a \in A(I)} r_I^+[a]} & \text{if } r_I[a] > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

[2]Perfect recall is required for convergence guarantees of the algorithms we mention in this paper. However, it has been shown in practice that relaxing this assumption can lead to large computational savings without significantly affecting the performance of the resulting strategy (Waugh et al. 2009) (Risk and Szafron 2010). Therefore, we will relax this assumption for our experiments on Texas Hold'em Poker.

where $r_I[a]$ is the cumulative sampled counterfactual regret of taking action a at I. If there is at least one positive regret, each action with positive regret is assigned a probability that is normalized over all positive regrets and the actions with negative regret are assigned probability 0. If all the regrets are negative, then the strategy is reset to a default uniform random strategy.

There are different ways to sample parts of the game tree. In this paper we will focus on **Outcome-Sampling MCCFR**. In outcome-sampling MCCFR $\mathcal{Q}$ is chosen so that each block contains a single terminal history, i.e., $\forall Q \in \mathcal{Q}, |Q| = 1$. On each iteration one terminal history is sampled and only updated each at information set along that history. The sampling probabilities, $\Pr(Q_j)$ must specify a distribution over terminal histories. MCCFR specifies this distribution using a *sampling profile*, σ', so that $\Pr(z) = \pi^{\sigma'}(z)$.

The algorithm works by sampling z using policy σ', storing $\pi^{\sigma'}(z)$. In particular, an ϵ-greedy strategy is used to choose the successor history: with probability ϵ choose uniformly randomly and probability $1 - \epsilon$ choose based on the current strategy. The single history is then traversed forward (to compute each player's probability of playing to reach each prefix of the history, $\pi_i^\sigma(h)$) and backward (to compute each player's probability of playing the remaining actions of the history, $\pi_i^\sigma(h, z)$). During the backward traversal, the sampled counterfactual regrets at each visited information set are computed (and added to the total regret). Here,

$$\tilde{r}(I,a) = \begin{cases} w_I(\pi^\sigma(z[I]a,z) - \pi^\sigma(z[I],z)) & \text{if } z[I]a \sqsubseteq z \\ -w_I \pi^\sigma(z[I],z) & \text{otherwise} \end{cases}$$

$$\text{where } w_I = \frac{u_i(z)\pi_{-i}^\sigma(z[I])}{\pi^{\sigma'}(z)} \qquad (5)$$

(Lanctot et al. 2009) provides a more in-depth discussion, including convergence proofs.

Monte Carlo Restricted Nash (MCRNR)

Regret minimization algorithms, as discussed in the previous section, learn an approximate NE strategy, i.e., the best possible worst-case strategy. Such a strategy treats opponents as rational players and is oblivious to opponent mistakes. When facing a predictable and inferior opponent, a tailored counter-strategy will earn more utility than a rational strategy. Given an accurate estimation (i.e. *model*) of the opponent strategy, one can compute a best-response strategy that maximally exploits it. However, opponent specific counter-strategies may be very brittle, and only perform well against the opponent they were trained against. (Johanson, Zinkevich, and Bowling 2008) empirically showed in the game of Poker that this was indeed the case. The authors also show that a NE strategy only wins by small margins, even against extremely exploitable opponents. The authors formulated a solution, namely a restricted Nash response that balances (1) performance maximization against the model and (2) reasonable performance even when the model is wrong.

RNR. More specifically, calculating the RNR response requires a model for the opponent. Suppose in a 2-player game, the opponent (i.e. restricted) player is player 2, then this model is $\sigma_{fix} \in \Sigma_2$. Define $\Sigma_2^{p,\sigma_{fix}}$ to be the set of mixed strategies of the form $p\sigma_{fix} + (1-p)\sigma_2$ where σ_2 is an arbitrary strategy in Σ_2. The set of restricted best responses to $\sigma_1 \in \Sigma_1$ is:

$$BR^{p,\sigma_{fix}}(\sigma_1) = \underset{\sigma_2 \in \Sigma_2^{p,\sigma_{fix}}}{\operatorname{argmax}} (u_2(\sigma_1, \sigma_2)) \qquad (6)$$

A (p, σ_{fix}) RNR equilibrium is a pair of strategies (σ_1^*, σ_2^*) where $\sigma_2^* \in BR^{p,\sigma_{fix}}(\sigma_1^*)$ and $\sigma_1^* \in BR(\sigma_2^*)$. In this pair, the strategy σ_1^* is a p-restricted Nash response to σ_{fix}. These are counter-strategies for σ_{fix}, where p provides a balance between exploitation and exploitability. Johanson, Zinkevich, and Bowling further proved that among all strategies that are at most ϵ-suboptimal, these strategies are among the best responses. They used the CFR algorithm to solve a modified game, wherein it is assumed that the opponent plays according to the fixed (model-provided) strategy with a certain probability, and according to a regret-minimizing strategy otherwise. Consequently, the algorithm requires a full sweep through the game tree.

MCRNR. In this paper, we extend the original RNR algorithm with sampling. The resulting new algorithm, MCRNR (for Sampled Restricted Nash), benefits from sampling only relevant parts of the game-tree. It is therefore able to converge very fast to robust best-response strategies. The pseudo-code of the algorithm is provided in Algorithm 1.

We will now explain the pseudo-code. We first assign a player, p_r, that is restricted to sometimes play using the fixed model. The SET_MODEL_CONFIDENCE routine returns a value in $[0, 1]$. The value represents the confidence of the model at the specific information set. In the original RNR paper, this is the value p used to trade-off exploitation for exploitability. When learning the model from data it makes sense to have different values per information set because the confidence depends on how many observations are available per information set; counter-strategies using a model built from data are called Data-Biased Responses (Johanson and Bowling 2009). We then sample a terminal history $h \in Z$, either selecting actions based on a provided opponent model, or based on the strategy obtained by regret-matching. The REGRET_MATCHING routine assigns a probability to an action in an information set (according to Equation 4). The sampling routine SELECT samples action a with probability $\epsilon/|A(I)| + (1-\epsilon)\sigma_i(I,a)$. At a terminal node, utilities are determined and backpropagated through all $z[I] \sqsubset z$.

Regret and average strategy updates are applied when the algorithm returns from the recursive call, in lines 17 to 22. On line 19 we add the sampled counterfactual regret (according to Equation 5) to the cumulative regret. On line 20 the average strategy is updated using *optimistic averaging*, which assumes that nothing has changed since the last visit at this information set. Finally, the strategy tables are updated before a new iteration starts. This process is repeated a number of times until satisfactory and there is one instance

initialize: Information set markers: $\forall I, c_I \leftarrow 0$
initialize: Regret tables: $\forall I, r_I[a] \leftarrow 0$
initialize: Strategy tables: $\forall I, s_I[a] \leftarrow 0$
initialize: Initial strategy: $\sigma(I, a) = 1/|A(I)|$
input : A starting history h
input : A sampling scheme $\mathcal{S}$ (ϵ-greedy)
input : An opponent model $\mathcal{M}$ for rest. player p_r
input : Current iteration t
Recursion Base Case:
if $h \in Z$ **then**
    $z \leftarrow h$
    return $(u_i(z), \pi^{\sigma'}(z), \pi^{\sigma}(z))$
Select Terminal Node:
$pConf \leftarrow$ SET_MODEL_CONFIDENCE(I_i)
$i \leftarrow P(h)$
if $(i = p_r)$ **and** $(rand(1) < pConf)$ **then**
    $\sigma_i \leftarrow \mathcal{M}(I_i)$
else
    $\sigma_i \leftarrow$ REGRET_MATCHING(r_{I_i})
$\sigma'_i \leftarrow \mathcal{S}(\sigma_i)$
$h' \leftarrow$ SELECT(h, σ'_i)
Recurse:
MCRNR$(h', \mathcal{S}, \mathcal{M}, t)$
Determine Utilities and Update:
$i \leftarrow P(h)$
foreach $a \in A(I)$ **do**
    $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$
    $s_I[a] \leftarrow s_I[a] + (t - c_I)\pi_i^{\sigma}\sigma_i(I, a)$
end
$c_I \leftarrow t$
return $(u_i(z), \pi^{\sigma'}(z), \pi^{\sigma}(z))$

Algorithm 1: One iteration of MCRNR

for each different player assigned to be the restricted player. At any iteration, the average strategy $\overline{\sigma}(I, a)$ can be obtained by normalizing s_I. When $p_r = 2$ then $\sigma_1^* = \overline{\sigma}_1$. When $p_r = 1$ then $\sigma_2^* = \overline{\sigma}_2$. Over time, $\sigma^* = (\sigma_1^*, \sigma_2^*)$ approaches an RNR equilibrium.

Experiments and Results

In this section we will discuss two sets of experiments with MCRNR in some smaller Poker-like games as well as in the large domain of Texas Hold'em Poker. We start by describing the games that the algorithm was applied to, and then the experiments themselves.

Goofspiel is a bidding card game where players have a hand of cards numbered 1 to N, and take turns secretly bidding on the top point-valued card in a point card stack using cards in their hands (Ross 1971). Our version is less informational: players only find out the result of each bid and not which cards were used to bid, and the player with the highest total points wins. We also use a fixed point card stack that is strictly decreasing, e.g. $(N, N-1, \ldots, 1)$.

Bluff(1,1,N) also known as Liar's Dice and Perudo, is a dice-bidding game. In our version, each player rolls a single N-sided die and looks at their die without showing it to their opponent. Then players, alternately, either increase the current bid on the outcome all die rolls in play or call the other player's bluff (claim that the bid does not hold). The highest value on the face of a die is wild and can count as any face value. When a player calls bluff, if the opponent's bid is incorrect they win, otherwise they lose.

One-Card Poker (abbreviated OCP(N)), is a generalization of Kuhn Poker (Gordon 2005) (Kuhn 1950). The deck contains N cards. Each player must ante a single chip, has one more chip to bet with, and is dealt one card. We use a deck of size $N = 500$.

Leduc Hold'em Poker (abbreviated LHP(R_1, R_2, B, i)), is a small game of Poker with a deck containing two suits of three cards each (Southey et al. 2005). Each player is dealt one card. There is a first round of betting with a raise amount of R_1 chips. Then a single community card is flipped, and there is a second round of betting with a raise amount of R_2 chips. There is a maximum number of raises per round B. The i parameter is a switch indicating whether raise amounts can be integers between 1 and R_n or must be exactly R_n. The standard Leduc Hold'em Poker has $(R_1, R_2, B, i) = (2, 4, 2, false)$.

Texas Hold'em Poker This variant of Poker is played between at least two players. Players can win games by either having the best card combination at the end of the game, or by being the only active player. The game includes four betting rounds wherein players are allowed to invest money. Players can remain active by at least matching the largest investment made by any of the players. This is known as *calling* or *checking*. Players may also decide to *bet* or *raise* a bet, which increases the stakes. Finally, they can choose to fold (i.e., stop investing money and forfeit the game). During the first betting round, all players are dealt two *private cards* that are only known to that specific player. During the remaining phases an additional 5 *board cards* appear that apply to all the players and are used to determine the card combinations.

Preliminary evidence

We ran two separate sets of experiments for each game except Texas Hold'em. The first set aimed to characterize the relationship between exploitation and exploitability for different values of $pConf$. The second set of experiments was a comparison of the convergence rates between the RNR and MCRNR. In both cases perfect opponent models were used taken from runs of MCCFR and ϵ was set to 0.6. Results are shown in Figures 1 and 2.

Results from the first set of experiments may influence the choice of $pConf$. If exploitation is much more important than exploitability then a value above 0.9 is suggested; on the other hand a noticeable boost in exploitation can be achieved for a small loss of exploitability for $0.5 \leq pConf \leq 0.8$. Unlike previous results, in every game except Bluff it seems that the region $pConf \in [0.97, 1]$ has high impact on to the magnitude of this trade-off. Results from Figure 2 confirm the performance benefit from sampling since MCRNR produces a better NE approximation in

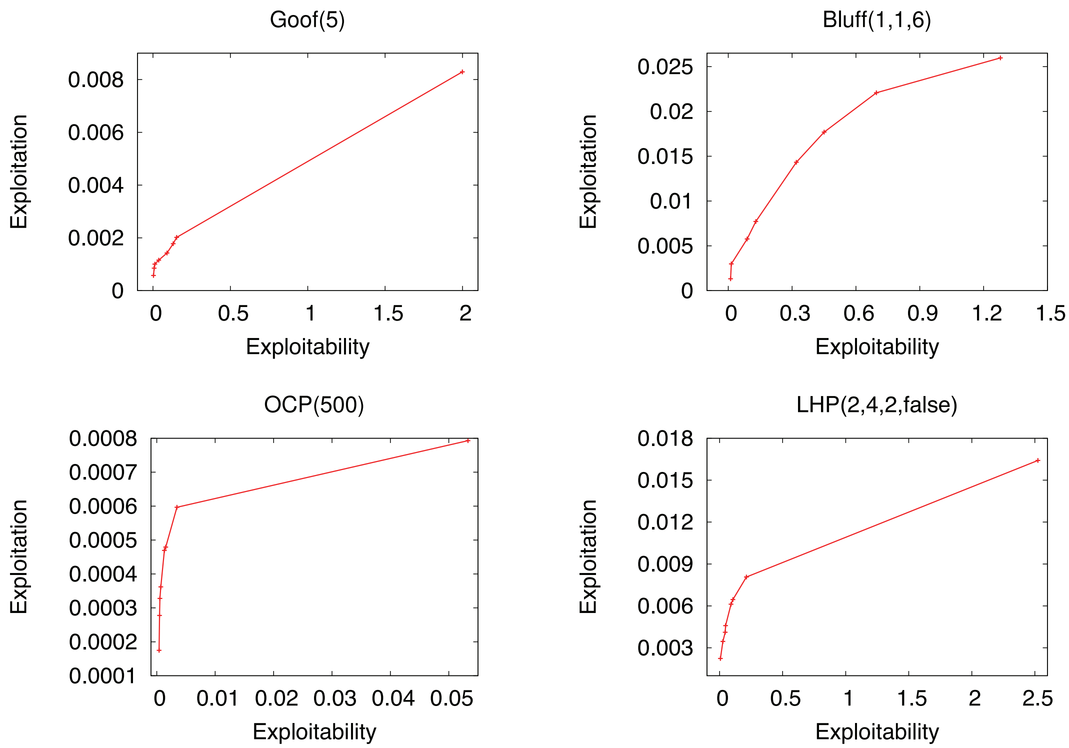

Figure 1: The trade-off between exploitation and exploitability. The exploitation value is the gain in payoff when using the MCRNR equilibrium profile vs. a Nash equilibrium profile, summed over $p_r \in \{1, 2\}$. The exploitability is $b_i(\sigma_{-i})$ summed over $i = p_r \in \{1, 2\}$. MCRNR was run for roughly three times the number of iterations as the MCCFR run that produced the opponent models. The value of $pConf$ used, from bottom-left point to top-right point, was: $0, 0.5, 0.7, 0.8, 0.9, 0.93, 0.97, 1$.

less time, especially in the early iterations. This is particularly important when attempting to learn online, when time might be limited.

Larger test domain: Texas Hold'em Poker

We evaluated the policies learned by MCCFR and MCRNR against two strong opponent bots provided with the software tool Poker Academy Pro, namely POKI and SPARBOT. Policies are evaluated in millibets per hand (mb/h), which describes the big blinds won per hand, divided by 1000. It can thus be used to reflect a player's (or bot's) playing skill. For a more detailed explanation of POKI and experiments with this bot, we refer the reader to (Billings 2006). SPARBOT is a bot that plays according to the NE strategy described by (Billings et al. 2003). It was designed solely for 2-player Poker, in contrast to POKI, which was designed for multiplayer games. Since SPARBOT specializes in 2-player games, it is less exploitable in a 2-player game than POKI.

To reduce the complexity of the task of finding NE and RNR strategies, we decreased the size of the game tree by applying a 10-bucket discretization (Billings 2006) on the cards, along with imperfect recall (i.e., buckets of previous phases are forgotten). At each phase, the strategic strength of private cards, along with zero or more board cards, determines the bucket.

We ran MCCFR using an epsilon-greedy sampling scheme. As suggested in earlier work (Lanctot et al. 2009), epsilon was set to the relatively high value of 0.6 to cover a large area of the search space.

For MCRNR we observed approximately $20K$ games played against both POKI and SPARBOT. These games were used to gather opponent data concerning the two bots. For our modeling, we used a a standard machine learning technique. Since the opponent data we gathered is rather sparse (e.g., (Johanson and Bowling 2009) used over 1 million games instead of $20K$) and since a frequency count cannot generalize, we chose to apply a standard decision tree to learn an opponent model from the sparse data. We provided the decision tree with five simple features, namely (1) the starting seat relative to the button, (2) the sum of bets or raises during the game, (3) the sum of bets or raises in the current phase, (4) the sum of bets or raises of the modeled player in the game, and finally (5) the bucket of the modeled player (if it was observed). For each specific phase, we learn a model that predicts the strategy of the modeled player. We set $pConf$ to a fixed value of 0.75. This value is not adapted based on the experience in a specific information set, as was done in the data-biased approach (Johanson and Bowling 2009).

We ran a number of offline iterations for MCCFR and MCRNR, froze the policy, and evaluated it. All results are shown in Table 1. We performed $10K$ evaluation games for

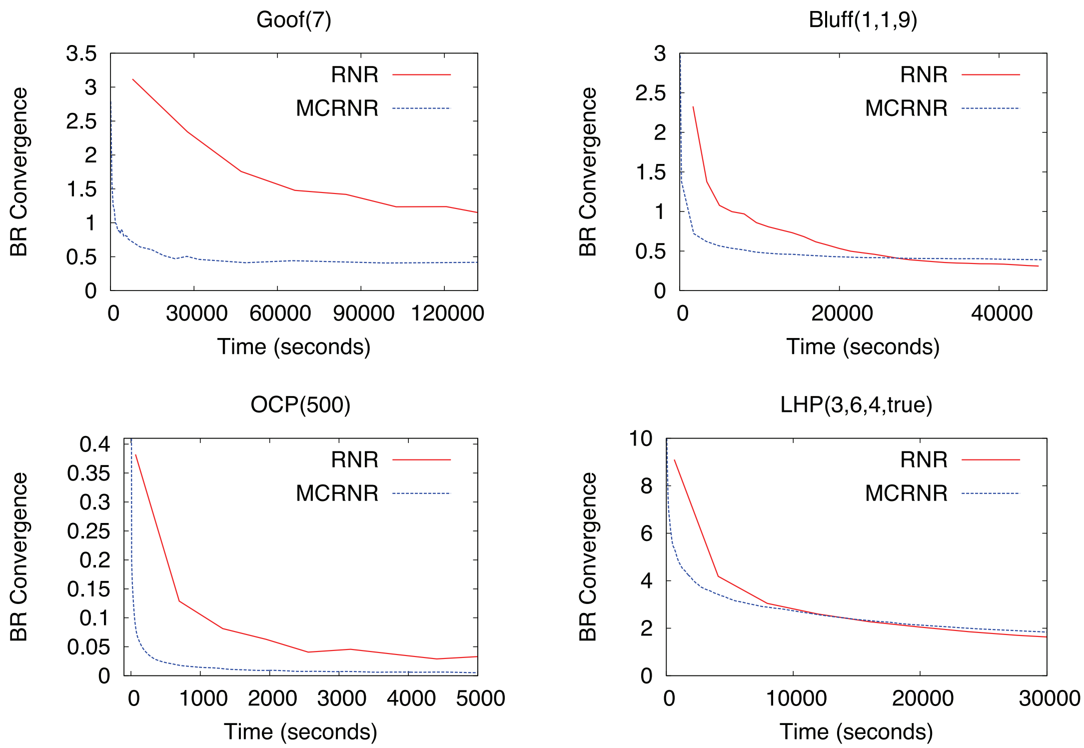

Figure 2: The convergence rates of RNR versus MCRNR. The y-axis is ϵ_σ defined in the Background section summed over $p_r \in \{1, 2\}$. When computing $b_{-i}(\sigma_i)$ and $i = p_r$, the restricted player uses $\sigma_{rest} = pConf \cdot \sigma_{fix} + (1 - pConf)\overline{\sigma}_i$. In these experiments, $pConf$ was set to 0.5.

each player. MCCFR, after a great deal of iterations, wins by a small margin from POKI, while losing a small amount from SPARBOT. We may conclude that MCCFR has learned a near-equilibrium policy.

As expected, MCRNR exploits POKI considerably more than MCCFR, namely with $369\,mb/h$. Interestingly, as depicted in Figure 3, MCRNR has learned to exploit POKI in about 20 million sampled iterations. We note that in a sampled iteration, only very few nodes are touched (i.e., only information sets are updated along the history of the sampled terminal node), while in a CFR (and RNR) iteration all information-sets are updated. Consequently, 20 million sampled iterations with MCCFR maps to far less iterations with CFR.

Against SPARBOT, an improvement is also observed, but here, the differences are not significant. It should be noted that both MCRNR policies were learned within only a fraction of the time (and number of iterations) required by MCCFR. We emphasize that MCRNR learned on the basis of only $20K$ observed games. In conclusion, MCRNR finds at least an equally good policy as MCCFR, but uses substantially less effort to do so.

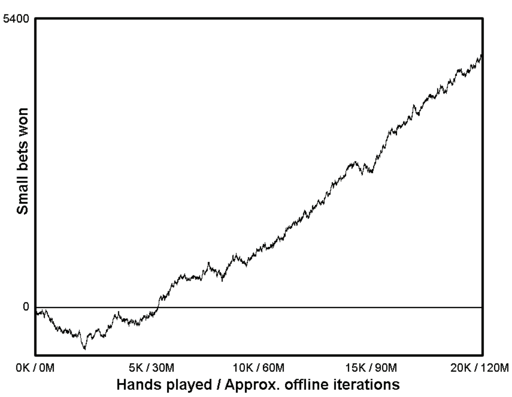

Figure 3: An online evaluation of MCRNR policy during learning against POKI. While the bot is playing $1,000$ online games an approximated 6 million offline iterations with MCRNR are run.

Conclusion

This paper presents MCRNR, a sample-based algorithm for the computation of restricted Nash strategies in complex extensive form games. This algorithm combines the advantages of two state-of-the-art existing algorithms, i.e., (Monte-Carlo) Counterfactual Regret Minimization and Restricted Nash. MCRNR samples only relevant parts of the game tree. It is therefore able to converge faster to robust best-response strategies than RNR.

Another advantage is that we can gather opponent data

Opponent	Algorithm	mb/h	Iterations
POKI	MCCFR	59	5400m
SPARBOT	MCCFR	-91	5400m
POKI	MCRNR	369	200m
SPARBOT	MCRNR	-39	200m

Table 1: Results of experiments with MCCFR and MCRNR against two bots . These results are significant to 60 mb/h. The fourth column gives the amount of offline iterations we allowed before stopping the algorithm.

online. (Johanson, Zinkevich, and Bowling 2008) gathered data using a 'Probe' bot, which either calls or bets with equal probability. This opponent makes sure that all parts of the tree are visited, and actions from each information set are observed. CFR and RNR make full sweeps through the tree, and require accurate statistics across the whole tree. In our case, sampling inherently provides this exploration measure, and the sample-corrected strategy updates boost the importance based on how long it has been since the last visit. So, when rarely-visited information sets are observed, their updates are more significant. Therefore MCRNR can learn models on-policy, which is faster (i.e., one gets more relevant observations for an opponent) and more practical (i.e., one can model opponents during play, running a 'probe' bot is not always possible).

We evaluate our algorithm on a variety of imperfect information games that are small enough to solve yet large enough to be strategically interesting, as well as a large game, Texas Hold'em Poker. In Poker MCRNR learns strong policies in a short amount of time. For future work we want to run more extensive experiments in the large domain of Poker. We want to gather more accurate statistics on the performance (robustness) of the algorithms. Furthermore, we are interested in an online application of the algorithm wherein we run MCRNR in the background while incrementally building an opponent model from observations.

References

Billings, D.; Burch, N.; Davidson, A.; Holte, R. C.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In Gottlob, G., and Walsh, T., eds., *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, 661–668. Morgan Kaufmann.

Billings, D. 2006. *Algorithms and Assessment in Computer Poker*. Ph.D. dissertation. University of Alberta.

Gordon, G. J. 2005. No-regret algorithms for structured prediction problems. Technical Report CMU-CALD-05-112, Carnegie Mellon University.

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

Johanson, M., and Bowling, M. 2009. Data biased robust counter strategies. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 264–271.

Johanson, M.; Zinkevich, M.; and Bowling, M. 2008. Computing robust counter-strategies. In *Advances in Neural Information Processing Systems 20 (NIPS)*.

Kuhn, H. W. 1950. Simplified two-person poker. *Contributions to the Theory of Games* 1:97–103.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 1078–1086.

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

Risk, N. A., and Szafron, D. 2010. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2010)*.

Ross, S. M. 1971. Goofspiel — the game of pure strategy. *Journal of Applied Probability* 8(3):621–625.

Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertaintyin Artificial Intelligence (UAI)*, 550–558.

Waugh, K.; Zinkevich, M.; Johanson, M.; Kan, M.; Schnizlein, D.; and Bowling, M. 2009. A practical use of imperfect recall. In *Proceedings of the 8th Symposium on Abstraction, Reformulation and Approximation (SARA)*.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20 (NIPS)*.