Search in Imperfect Information Games Using Online Monte Carlo Counterfactual Regret Minimization

Marc Lanctot[1], Viliam Lisý[2], and Michael Bowling[3]

[1]Department of Knowledge Engineering, Maastricht University, The Netherlands
[2]Department of Computer Science, FEE, Czech Technical University in Prague
[3]Department of Computing Science, University of Alberta, Canada
marc.lanctot@maastrichtunviersity.nl, viliam.lisy@agents.fel.cvut.cz, bowling@cs.ualberta.ca

Abstract

Online search in games has always been a core interest of artificial intelligence. Advances made in search for perfect information games (such as Chess, Checkers, Go, and Backgammon) have led to AI capable of defeating the world's top human experts. Search in imperfect information games (such as Poker, Bridge, and Skat) is significantly more challenging due to the complexities introduced by hidden information. In this paper, we present Online Outcome Sampling (OOS), the first imperfect information search algorithm that is guaranteed to converge to an equilibrium strategy in two-player zero-sum games. We show that OOS avoids common problems encountered by existing search algorithms and we experimentally evaluate its convergence rate and practical performance against benchmark strategies in Liar's Dice and a variant of Goofspiel. We show that unlike with Information Set Monte Carlo Tree Search (ISMCTS) the exploitability of the strategies produced by OOS decreases as the amount of search time increases. In practice, OOS performs as well as ISMCTS in head-to-head play while producing strategies with lower exploitability given the same search time.

Introduction

In many sequential multi-agent interactions, agents have some initial time to prepare for the interaction and then after each decision, have additional thinking time to decide about their next move. When preparation time is abundant and the computational resources are sufficient, an equilibrium strategy for a smaller abstract game can be precomputed and then used during the game play. This *offline approach* has been remarkably successful in Computer Poker (Sandholm 2010; Rubin and Watson 2010; Gilpin 2009; Johanson 2007). However, the preparation time is often very limited. The exact model of the game may become known only shortly before acting is necessary, such as in general game-playing, security enforcement in a previously unknown environment, and general-purpose robotics. In a short time, only a very small abstract game could be solved in advance. Moreover, it might not even be possible to create a sufficiently small and still useful abstraction to be solved in time. In these cases, agents may need to *decide online*: make initial decisions quickly and then put additional effort to improving their strategy in the current situation while the interaction is taking place.

Some of the first imperfect information search algorithms were inspired by the application to Bridge (Frank, Basin, and Matsubara 1998; Ginsberg 1996), and notably by the success of GIB (Ginsberg 2001). In GIB, perfect information search is performed on a *determinized* instance of the game: one where all players can see usually hidden information, a concept originally proposed in (Levy 1989). This approach has also performed well in other games like Scrabble (Sheppard 2002), Hearts (Sturtevant 2008), and Skat (Buro et al. 2009).

There are several problems that have been identified with Perfect Information Monte Carlo (PIMC) search techniques on determinized samples. The most common are *strategy fusion* and *non-locality* (Frank, Basin, and Matsubara 1998). Strategy fusion occurs when an algorithm is allowed to choose two different actions from two different states in the same information set, which is inconsistent with the constraints imposed by the game's information structure. Non-locality occurs due to the assumption that subgames are well-defined and that hence search can be recursively applied by computing and comparing values of subgames.

Strategy fusion can be overcome by imposing the proper information constraints during search, as was done in Kriegspiel (Ciancarini and Favini 2010), Kuhn Poker (Ponsen, de Jong, and Lanctot 2011), pursuit-evasion games (Lisy, Bosansky, and Pechoucek 2012), card games and other phantom games (Cowling, Powley, and Whitehouse 2012). Non-locality, however, is a fundamental problem that prevents the usual minimax formulation of optimal payoffs defined strictly over children subtrees. There is empirical evidence suggesting that some of these techniques will not converge to the optimal solution even in very small games like Biased Rock-Paper-Scissors and Kuhn poker (Shafiei, Sturtevant, and Schaeffer 2009; Ponsen, de Jong, and Lanctot 2011).

Game-tree search techniques have also been applied to Poker (Billings et al. 2004). In recent years, however, the most common approach to producing strong Poker AI has been use advances in equilibrium approximation algorithms to compute a near-optimal equilibrium of an abstract game and subsequently use the abstract strategies when playing the full game (Sandholm 2010). We refer to this as the *offline approach* because it requires designing domain-specific ab-

stractions and precomputing approximate equilibria offline in advance, which could take several weeks or even months. In the offline approach, during actual play an agent simply looks up its precomputed strategy stored in a large table. This offline approach is often not possible. For example, an agent may be required to act shortly after discovering the rules, such as in general game playing, which leaves very little time for precomputation of equilibria or designing of domain-specific abstractions. Also, due to the size of the abstractions required for strong play (Johanson et al. 2013), there may not be enough space to store the precomputed strategy, such as in gaming apps on mobile devices.

In this paper, we focus on the *online* setting. We propose a Monte Carlo Tree Search (MCTS) (Browne et al. 2012) algorithm that uses Monte Carlo Counterfactual Regret Minimization as its basis rather than Upper Confidence Bounds (Auer, Cesa-Bianchi, and Fischer 2002; Kocsis and Szepesvári 2006). We introduce Online Outcome Sampling (OOS), a simulation-based algorithm that builds its search tree incrementally, like MCTS. We show that OOS converges to an equilibrium strategy as search time per move increases, making it the first known imperfect information search algorithm satisfying this property, which we call *consistency*. We show the empirical convergence rates and performance of OOS, comparing them to benchmark players and to ISMCTS (Cowling, Powley, and Whitehouse 2012), a recent algorithm currently used in a popular mobile phone implementation of Spades (Whitehouse et al. 2013).

Extensive-Form Games

Here, we define the relevant game-theoretic terminology that forms the basis of our analysis. The notation used here is based on (Osborne and Rubinstein 1994).

In this paper, we focus on two-player zero-sum extensive form games, which model sequential decision making of agents called **players** denoted $i \in N = \{1, 2\}$. In turn, players choose **actions** leading to sequences called **histories** $h \in H$. A history $z \in Z$, where $Z \subseteq H$, is called a **terminal history** and represents a full game from start to end. At each terminal history z there is a payoff $u_i(z)$ in $[-\Delta, \Delta]$ to each player i. At each nonterminal history h, there is a single current player to act, determined by $P : H \backslash Z \to N \cup \{c\}$ where c is a special player called **chance** (sometimes also called nature) that plays with a fixed stochastic strategy. For example, chance is used to represent rolls of dice and card draws. The game starts in the empty history $\emptyset$, and at each step, given the current history h, the current player chooses an action $a \in A(h)$ leading to successor history $h' = ha$; in this case we call h a **prefix** of h' and denote this relationship by $h \sqsubset h'$. Also, for all $h, h', h'' \in H$, if $h \sqsubset h'$ and $h' \sqsubset h''$ then $h \sqsubset h''$, and $h \sqsubset h$. Each set N, H, Z, and $A(h)$ for every $h \in H$ are finite and every history has finite length.

Define $\mathcal{I} = \{\mathcal{I}_i \mid i \in N\}$ the set of information partitions. $\mathcal{I}_i$ is a partition over $H_i = \{h \mid P(h) = i\}$ where each part is call an **information set**. Intuitively, an information set $I \in \mathcal{I}_i$ that belongs to player i represents a state of the game with respect to what player i knows. Formally, I is a set of histories that a player cannot tell apart (due to information hidden from that player). For all $h, h' \in I$, $A(h) = A(h')$ and $P(h) = P(h')$; hence, often we naturally extend the definition to $A(I)$, $P(I)$, and denote $I(h)$ the information set containing h.

A **behavioral strategy** for player i is a function mapping each information set $I \in \mathcal{I}_i$ to a probability distribution over the actions $A(I)$, denoted by $\sigma_i(I)$. Given a profile σ, we denote the probability of reaching a terminal history z under σ as $\pi^\sigma(z) = \prod_{i \in N} \pi_i(z)$, where each $\pi_i(z) = \prod_{ha \sqsubset z, P(h)=i} \sigma_i(I(h), a)$ is a product of probabilities of the actions taken by player i along z. We also use $\pi^\sigma_i(h, z)$ and $\pi^\sigma(h, z)$ to refer to the product of only the probabilities of actions along the sequence from the end of h to the end of z, where $h \sqsubset z$. Define Σ_i to be the set of behavioral strategies for player i. As is convention, σ_{-i} and π^σ_{-i} refer to player i's opponent strategy and products of the opponent's and chance's actions. An **ϵ-Nash equilibrium**, σ, is a set of σ_i for $i \in N$ such that the benefit of switching to some alternative σ'_i is limited by ϵ, i.e.,

$$\max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i}) - u_i(\sigma) \leq \epsilon \tag{1}$$

holds for each player $i \in N$. When $\epsilon = 0$, the profile is simply called a Nash equilibrium. When $|N| = 2$ and $u_1(z) + u_2(z) = k$ for all $z \in Z$, then the game is a two-player zero-sum game; these games form an important subset of extensive-form games due to their worst-case guarantees: different equilibrium strategies result in the same expected payoff against any arbitrary opponent equilibrium strategy and at least the same payoff for any opponent strategy at all. In this paper, we define the **exploitability** of a profile to be the sum of both distances from Eq. 1, $\epsilon_\sigma = \max_{\sigma'_1 \in \Sigma_1} u_1(\sigma'_1, \sigma_2) + \max_{\sigma'_2 \in \Sigma_1} u_2(\sigma_1, \sigma'_2)$.

Counterfactual Regret Minimization

Counterfactual Regret (CFR) is a notion of regret at the information set level for extensive-form games (Zinkevich et al. 2008). The CFR algorithm iteratively learns strategies in self-play, converging to an equilibrium. The **counterfactual value** of reaching information set I is the expected payoff given that player i played to reach I, the opponents played σ_{-i} and both players played σ after I was reached:

$$v_i(I, \sigma) = \sum_{(h,z) \in Z_I} \pi^\sigma_{-i}(h) \pi^\sigma_i(h, z) u_i(z), \tag{2}$$

where $Z_I = \{(h, z) \mid z \in Z, h \in I, h \sqsubset z\}$. Suppose, at time t, player i plays with strategy σ^t_i. Define $\sigma^t_{I \to a}$ as identical to σ^t except at I action a is taken with probability 1. The counterfactual regret of not taking $a \in A(I)$ at time t is $r^t(I, a) = v_i(I, \sigma^t_{I \to a}) - v_i(I, \sigma^t)$. The algorithm maintains the cumulative regret $R^T(I, a) = \sum_{t=1}^T r^t(I, a)$, for every action at every information set of every player. Then, the distribution at each information set for the next iteration $\sigma^{T+1}(I)$ is obtained individually using regret-matching (Hart and Mas-Colell 2000). The distribution is proportional to the positive portion of the individual actions' regret:

$$\sigma^{T+1}(I, a) = \begin{cases} R^{T,+}(I, a) / R^{T,+}_{sum}(I) & \text{if } R^{T,+}_{sum}(I) > 0 \\ 1/|A(I)| & \text{otherwise,} \end{cases}$$

where $x^+ = \max(0, x)$ for any term x, and $R^{T,+}_{sum}(I) = \sum_{a' \in A(I)} R^{T,+}(I, a')$. Furthermore, the algorithm maintains for each information set the average strategy profile

$$\bar{\sigma}^T(I, a) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(I, a)}{\sum_{t=1} \pi_i^{\sigma^t}(I)}, \quad (3)$$

where $\pi_i^{\sigma^t}(I) = \sum_{h \in I} \pi_i^{\sigma^t}(h)$. The combination of the counterfactual regret minimizers in individual information sets also minimizes the overall average regret (Zinkevich et al. 2008), and hence the average profile is a 2ϵ-equilibrium, with $\epsilon \to 0$ as $T \to \infty$.

Monte Carlo Counterfactual Regret Minimization (MCCFR) applies CFR to sampled portions of the games (Lanctot et al. 2009). In the **outcome sampling** (OS) variant of the algorithm, a single terminal history $z \in Z$ is sampled in each iteration. The algorithm updates the regret in the information sets visited along z using the **sampled counterfactual value**,

$$\tilde{v}_i(I, \sigma) = \begin{cases} \frac{1}{q(z)} \pi^{\sigma}_{-i}(z)\pi^{\sigma}_i(h, z)u_i(z) & \text{if } (h, z) \in Z_I \\ 0 & \text{otherwise,} \end{cases}$$

where $q(z)$ is the probability of sampling z. As long every $z \in Z$ has non-zero probability of being sampled, $\tilde{v}_i(I, \sigma)$ is an unbiased estimate of $v(I, \sigma)$ due to the importance sampling correction $(1/q(z))$. For this reason, applying CFR updates using these sampled counterfactual values on the sampled information sets values also eventually converges to the approximate equilibrium of the game with high probability. The required number of iterations for convergence is much larger, but each iteration is much faster.

In Poker, CFR and MCCFR have been used with remarkable success as offline methods for pre-computing approximate equilibria in abstract games (Zinkevich et al. 2008; Johanson et al. 2012a); the same general approach has also been used in Liar's Dice (Neller and Hnath 2011; Lanctot et al. 2012).

Subgames and Online Search

In a **match** (online game), each player is allowed little or no preparation time before playing (preventing the offline advance computation of approximate equilibria solutions). There is a current **match history**, $\mathtt{h}$, initially the empty history $\emptyset$ representing the start of the match. Each turn, the agent controlling $P(\mathtt{h})$ is given t time units to decide on a **match action** $\mathtt{a} \in A(\mathtt{h})$ and the match history then changes using $\mathtt{h} \leftarrow \mathtt{ha}$. There is a single referee who knows $\mathtt{h}$, samples chance outcomes as needed from $\sigma_c(\mathtt{h})$, and reveals $I(\mathtt{h})$ to $P(\mathtt{h})$ on their turn. The players play until the match is terminated, giving each player i a payoff of $u_i(\mathtt{z})$.

A perfect information game can be broken down into subgames and solved independently. Every perfect information game has a pure subgame perfect equilibrium, which can be found by backward induction. Search algorithms simply aim to identify the single optimal action at the search tree's root. Imperfect information games cannot be easily broken down into subgames. In general, the optimal strategies in an inner information set could be mixed and depend on the probabilities that individual nodes in the information set are reached.

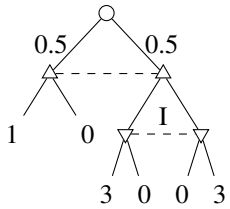

Figure 1: An example game with maximizing $\triangle$, minimizing $\triangledown$ and chance $\bigcirc$ players. The optimal (Nash equilibrium) strategy is for $\triangle$ to play (0.5,0.5) and for $\triangledown$ to play (left,right) with probabilities $(\frac{1}{3}, \frac{2}{3})$. Its value is 1.0.

These probabilities depend on the strategies of the players in the tree above the information set, which in turn depend on the payoffs obtained from other parts of the tree.

Some previous work has been done for accelerating equilibrium computation by solving or approximately solving subgames, particularly in end-game situations (Gilpin and Sandholm 2006; 2007; Ganzfried and Sandholm 2013). These techniques tend to help in practice, but as shown in (Ganzfried and Sandholm 2013), non-locality prevents these methods from producing equilibrium strategies in general. Using CFR to solve decomposed subgames has also recently been investigated (Burch, Johanson, and Bowling 2014). Whether this decomposition could be used in Monte Carlo simulations is an interesting topic of future work.

Information Set Monte Carlo Tree Search

The implementation details of ISMCTS described in (Cowling, Powley, and Whitehouse 2012) have impact on efficiency of the method for various games, but in essence, the multi-observer variant of the algorithm incrementally builds the game tree similarly to MCTS, but it places a multi-armed bandit algorithm at each information set rather than at each state. After a non-root information set is reached during the match, further simulations are run from a random node in this information set with uniform probability. This prevents the algorithm from converging to the optimal solution in the game. Consider the game in Figure 1. Suppose ISMCTS searches from information set I. Because utilities for actions taken from both states are combined, ISMCTS will choose left and right action equally often. However, mixing uniformly at I is not part of an equilibrium in this game, since it would lead to an expected utility of $\frac{3}{4}$ to the maximizing player for playing right, which would give an incentive to the maximizer to always play left and in that case, the minimizer would be better of playing right, reaching expected reward of 0.5.

Online Outcome Sampling

When outcome sampling is used in the offline setting, data structures for all information sets are allocated and created before the first iteration starts. In each iteration, every information set that is sampled gets updated.

We make two essential modifications to adapt outcome sampling to the online search setting.

Incremental Game Tree Building. Before the match begins, only the very first (root) information set is added to memory. In each iteration, a single information set (at most) is added the information set tree (memory) each iteration. In particular, when an information set is reached that is not in memory, it is added to memory and then a default playout policy (*e.g.* uniform random) takes over for the remainder of the simulation. Along the playout portion (tail) of the simulation, information sets are not added to memory nor updated. Along the tree portion (head) of simulation, information sets are updated as normal. This way, only the information sets with relevant statistics will be stored in the memory.

In-Match Search Targeting. Suppose several moves have been played since the start of the match leading to $\mathtt{h}$. Plain outcome sampling would continue to sample from the root of the game (not the current match history $\mathtt{h}$), entirely disregarding the region of the game space that the match has headed toward. Hence, the second modification we propose is directing the search towards the histories that are more likely to occur during the match currently played. Note that this history is typically unknown to the players, who know only their information sets it leads to. Furthermore, unlike in ISMCTS, OOS always runs samples form the root of the game tree, even with non-empty match history.

We now describe two specific targeting methods.

Information Set Targeting (IST)

Suppose the match history is $\mathtt{h}$. IST samples histories reaching the current information set ($I(\mathtt{h})$), i.e., $(h, z) \in Z_{I(\mathtt{h})}$, with higher probability than other histories. The intuition is that these histories are particularly relevant since the searching player *knows* that one of these z will describe the match at its completion. However, focusing fully only on these histories may cause problems, since convergence guarantees are lost. Consider again the game in Figure 1. If the minimizing player knows it is in the information set I and focuses all its search only to this information set for sufficiently long, she computes the uniform strategy, which is optimal in the right (coordination) part of the game. However, if the minimizing player plays uniformly, the maximizing player prefers to switch to always play the left action to increase its payoff in case of not playing the coordination game. Any fixed non-zero probability of sampling the left chance action will eventually solve the problem. The regrets are multiplied by the reciprocal of the sampling probability; hence, they influence the strategy in the information set proportionally stronger if the samples are rare.

Note that previous methods, such as PIMC and ISMCTS, *always* target $I(\mathtt{h})$, *i.e.* with probability 1, and do not update predecessors of $I(\mathtt{h})$. In contrast, in IST *all* information sets in memory reached during each iteration requires updating to ensure eventual convergence to an equilibrium.

Public Subgame Targeting (PST)

A **public action** is an action in the "public tree" defined in (Johanson et al. 2012b). Informally, an action is said to be public if it is observable by all players (e.g., a bid in Liar's Dice or Poker is public). Formally, an action a is public, iff $\forall i, \forall I \in \mathcal{I}_i, \forall h_1, h_2 \in I : a \in h_1 \Leftrightarrow a \in h_2$. For example, the extensive-form version of Rock, Paper, Scissors has two information sets $I_1 = \emptyset$ and $I_2 = \{r, p, s\}$; it has no public actions, because each history in I_2 contains a single unique action (the unobserved ones taken by the first player).

Given a history h, let $p(h)$ be the sequence of public actions along h in the same order that they were taken in h. Define the **public subgame** induced by I to be the one whose terminal history set is

$$Z_{p,I(h)} = \{(h', z) \mid z \in Z, h' \in H, p(h') = p(h), h' \sqsubset z\}.$$

Now, suppose the match history is $\mathtt{h}$. Public subgame targeting samples $z \in Z_{p,I(\mathtt{h})}$ with higher probability than terminal histories outside this set.

A public subgame then, contains all the terminal histories consistent with the bidding sequence played over the match and each combination of private chance events for both players. So, in a game of two-player limit Texas Hold'em poker, suppose first player bets and second player calls, and then flop is revealed. At this point, the public actions are: bet, call. The public subgame described by $Z_{p,I(h)}$ contains every terminal history (including every combination of private chance outcomes for all players) with at least two public actions, whose first two public actions are: bet, call.

Algorithm

The algorithm is iterative and samples a single trajectory from the root $\emptyset$ to some terminal history. At each information set in memory, I, there are two tables maintained: r_I stores cumulative regret for each action $a \in A(I)$, and s_I stores the cumulative average strategy probability of each action.

Depending on the targeting method that is chosen (IST or PST), Z_{sub} is one of $Z_{I(\mathtt{h})}$ or $Z_{p,I(\mathtt{h})}$. The pseudo-code is presented as Algorithm 1. Each iteration is represented by two calls of OOS where the update player $i \in \{1, 2\}$ is alternated. Before each iteration, a *scenario* is decided: with probability δ the iteration targets the subgame and chooses $z \in Z_{sub}$ and with probability $(1-\delta)$ the usual OS sampling determines $z \in Z$. The first parameter of OOS is the current history. The next two are strategy's reach probabilities for the update player i and the opponent. The third and fourth parameters are overall probabilities that the current sample is generated, one for each scenario: first the targeted and then the untargeted. The last is the update player. Initial calls have the form $\text{OOS}(\emptyset, 1, 1, 1, 1, i)$. For the return values, x is a suffix/tail reach probability for both players, l is the root-to-leaf sample probability, and u is the payoff of the trajectory in view of the update player.

In outcome sampling, an ϵ-on-policy sampling distribution used at each information set is defined as

$$\Phi(I, i) = \begin{cases} \epsilon \cdot \text{Unif}(A(I)) + (1 - \epsilon)\sigma_i(I) & \text{if } P(I) = i \\ \sigma_i(I) & \text{otherwise,} \end{cases}$$

and denote $\Phi(I, i, a)$ the probability of sampling $a \in A(I)$.

The sampling at chance's choices on line 5 depends on the method and the scenario being used. For example, when using information set targeting, the outcome that is sampled must be consistent with match history.

1 OOS$(h, \pi_i, \pi_{-i}, s_1, s_2, i)$:
2 **if** $h \in Z$ **then**
3 **return** $(1, \delta s_1 + (1-\delta)s_2, u_i(z))$
4 **else if** $P(h) = c$ **then**
5 Sample an outcome a and let ρ_1, ρ_2 be its probability in targeted and untargeted setting
6 **return** OOS$(ha, \pi_i, \rho_2\pi_{-i}, \rho_1 s_1, \rho_2 s_2, i)$
7 $I \leftarrow$ getInfoset$(h, P(h))$
8 Let $(a, s_1', s_2') \leftarrow$ Sample(h, I, i, ϵ)
9 **if** I *is not in memory* **then**
10 Add I to memory
11 $\sigma(I) \leftarrow$ Unif$(A(I))$
12 $(x, l, u) \leftarrow$ Playout$(ha, \delta s_1 + (1-\delta)s_2)$
13 **else**
14 $\sigma(I) \leftarrow$ RegretMatching(r_I)
15 $\pi'_{P(h)} \leftarrow \sigma(I, a)\pi_{P(h)}$
16 $\pi'_{-P(h)} \leftarrow \pi_{-P(h)}$
17 $(x, l, u) \leftarrow$ OOS$(ha, \pi_i', \pi'_{-i}, s_1', s_2', i)$
18 $c \leftarrow x$
19 $x \leftarrow x\sigma(I, a)$
20 **for** $a' \in A(I)$ **do**
21 **if** $P(h) = i$ **then**
22 $W \leftarrow u\pi_{-i}$ / l
23 **if** $a' = a$ **then**
24 $r_I[a'] \leftarrow r_I[a'] + (c - x)W$
25 **else**
26 $r_I[a'] \leftarrow r_I[a'] - xW$
27 **else**
28 $s_I[a'] \leftarrow s_I[a'] + \frac{1}{\delta s_1 + (1-\delta)s_2}\pi_{-i}\sigma(I, a')$
29 **return** (x, l, u)

Algorithm 1: Online Outcome Sampling.

A critical part of the algorithm is the action chosen and sample reach updates on line 8. In the targeted scenario, the current history h is always in the targeted part of the game and an action from $\{a \mid \exists z \in Z \ (ha, z) \in Z_{sub}\}$ is selected using the distribution $\Phi(I(h), i)$ normalized to one on this subset of actions. If we define $sum = \sum_{(ha,z)\in Z_{sub}} \Phi(I, i, a)$ then $s_1' = s_1\Phi(I, i, a)/sum$. In the untargeted scenario, any action $a \sim \Phi(I, i)$ can be sampled. If the action is not leaving the targeted part of the game (i.e., $(ha, z) \in Z_{sub}$) then $s_1' = s_1\Phi(I, i, a)/sum$ otherwise $s_1' = 0$. In all cases $s_2' = \Phi(I, i, a)s_2$.

These sample reach probabilities are combined into one true reach probability at a terminal history on line 2, starting the playout on line 12 and when updating the average strategy on line 28.

The playout on line 12 samples to the end of the game with some playout policy at each step; we use uniform random, but in general one could use a informed policy based on domain knowledge as well. Unlike MCTS, the playout policy in OOS must compute l when reaching a terminal and update the tail probability x when returning as done on line 19. Lines 15 and 16 simply modify the $P(h)$'s reach probability by multiplying it by $\sigma(I, a)$, keeping the value of the other one the same.

Lines 18 to 24 contain the usual outcome sampling updates. Note that regrets are updated at the update player histories, while average strategy tables at opponent histories.

Theorem 1. *Let $\bar{\sigma}^t_m(\delta, \mathtt{h})$ be a strategy produced by OOS with scheme $m \in \{IST, PST\}$ using $\delta < 1$ started from $\mathtt{h}$ run for t iterations, with exploration $\epsilon > 0$. For any $p \in (0, 1], \varepsilon > 0$ there exists $t < \infty$ such that with probability $1 - p$ the strategy $\bar{\sigma}^t_m(\delta, \mathtt{h})$ is a ε-equilibrium strategy.*

Since every terminal history has non-zero probability of being sampled, eventually every information set will be contained in memory. Then, the algorithm becomes MCCFR with a non-uniform sampling scheme. Consequently by (Lanctot et al. 2009, Theorem 5) OOS minimizes external regret with high probability.

Note that due to non-locality, this consistency property cannot hold generally for any search algorithm that does not modify $\sigma(I)$ at previous $I(h)$ such that $h \sqsubset \mathtt{h}$. However, it is an open question as to whether any of the previous algorithms could be modified to ensure consistency.

Empirical Evaluation

We now compare the head-to-head performance and exploitability of OOS and ISMCTS on two games.

Liar's Dice, LD(D_1,D_2), also known as Dudo, Perudo, and Bluff is a dice-bidding game. Each die has six sides with faces ⚀ to ⚅ and a star ⋆. Each player i rolls D_i of these dice and looks at them without showing them to their opponent. Each round, players alternate by bidding on the outcome of all dice in play until one player "calls liar", *i.e.* claims that their opponent's latest bid does not hold. A bid consists of a quantity of dice and a face value. A face of ⋆ is considered wild and counts as matching any other face. To bid, the player must increase either the quantity or face value of the current bid (or both). The losing player discards a number of dice equal to how many dice were missing to have a valid bid. The players continue until one player has no more dice.

Imperfect Information II-Goofspiel(N) is a two-player card game where each player is given a private hand of bid cards with values 1 to N. A different deck of N point cards is placed face up in a stack On their turn, each player bids for the top point card by choosing a single card in their hand. The highest bidder gets the point card and adds the point total to their score, discarding the points in the case of a tie. This is repeated N times and the winner is the player with the highest score. In, II-Goofspiel the players only discover who won or lost a bid but not the bid cards played. Also, we assume the point-stack is strictly increasing: 1, 2, ... N.

We will focus our experiments on LD(1,1) and II-Goofspiel(6). While these games are considered small by search algorithm standards, it is still possible to compute best response strategies to measure exploitability, allowing us to show the observed convergence of the strategies produced by OOS. We include preliminary results in LD(2,2).

To improve performance against irrational play, we use a more explorative regret matching, $\sigma^{T+1}_\gamma(I, a) = \gamma/|A(I)| + (1-\gamma)\sigma^{T+1}(I, a)$, with $\gamma = 0.01$. While this could effect convergence, we observe in our experiments that exploitability decreases as search time increases.

Head-to-Head Performance versus Exploitability

In games like Poker and Liar's Dice, it is often critical to play in such a way that the opponent cannot easily infer the private information. This explains partly why CFR-based methods have enjoyed so much success in the offline approach. In the online setting, however, since the tree is built incrementally, only partial strategies are produced. We are unaware of any methods for assessing the worst-case exploitability of strategies produced by an online search algorithm. We therefore propose two new methods to approximate the exploitability the produced strategies.

In the offline setting, measuring exploitability can be done by a recursive walk of the game tree using expectimax. In online search, the algorithm computes only a partial strategy. The first **full stitching** method enumerates each $I \in \mathcal{I}_i$ in topologically-sorted order starting at the root, running a search from I re-using only the information computed in previous searches from ancestors of I, saving the distribution computed at I, and passing down the state of memory only for children of I. We do not save changes made to ancestors when searching at I to ensure a fair comparison between OOS and ISMCTS. Full stitching provides the best representation of a full strategy that would eventually be produced by OOS since it builds distributions at each information set in the same way as OOS would if they were reached during play. However, full-stitching requires $|\mathcal{I}|$ searches and memory, which is impractical on large games.

We also propose a the multi-match **aggregate method**. This method first creates a global (accumulating) strategy data structure for each player type and generates a set of matches M. Then, each $m \in M$ is simulated invoking the appropriate search algorithm at each observed I along m. Since m is predetermined, the choice made by the search algorithm is discarded, but the information computed (visit counts in ISMCTS, s_I in OOS) is added into the global strategy data structure belonging to the player who searched. For a fair comparison, the first I reached along each m aggregates all the information gained in the search but for future I', only the information collected in each I'' reachable by I' is aggregated. Note that it is safe to combine the information in this way: in ISMCTS the actions chosen and visits are independent of how I' was reached. In OOS, the accumulating s_I values of two converging ϵ-equilibrium average strategies can be combined due to linearity of expected utility.

In our experiments ISMCTS uses $C = 2$; tuning shows that the C value does not affect the performance. In II-Goofspiel(6) we run searches for 1, 2, and 4 seconds. As there are no public actions, we only compare IST in II-Goofspiel. Over a range of values for δ and all time settings, there was no statistically significant winner in head-to-head matches between IST and ISMCTS. Exploitability is shown in Figure 2. For reference, the exploitability values of ISMCTS was 0.95, 0.96, 0.91 for search times of 1, 2, and 4 seconds. We observe that exploitability generally decreases as search time increases and and changes little as δ increases, with its lowest points at $\delta = 1$. This result was surprising, and we suspect it is due OOS benefiting from the reuse of values from previous searches in the same match.

In LD(1,1), we try searches for 1, 5, and 25 seconds.

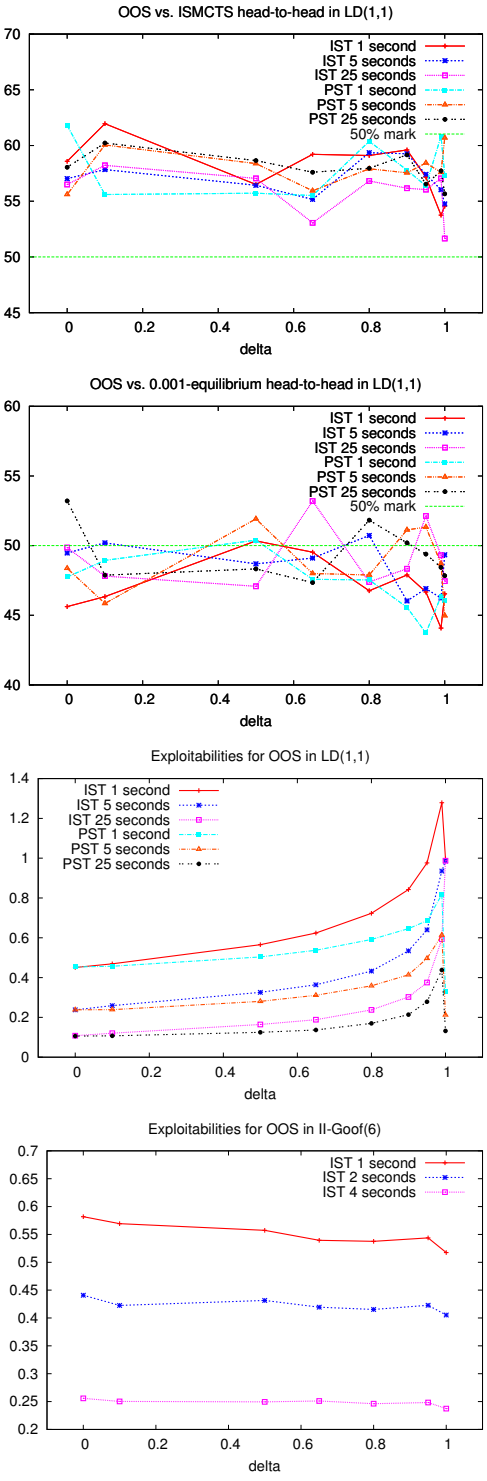

Figure 2: Results for OOS in LD(1,1). From top: win rate (%) of OOS vs. ISMCTS, win rate (%) of OOS vs. a 0.001-equilibrium, approximate ϵ_σ using aggregate method in LD(1,1) and II-Goofpiel(6).

When played against a uniform random player, among all values for delta and time settings, IST wins 79.2-86.7% and PST wins 78.6-84.0%, with a non-noticeable differences across δ values. ISMCTS beat uniform random 79.8-82.4% at 1, 5, and 25 seconds. Upon further inspection, ISMCTS very quickly converges to the same strategy every time: as first player, with a weak roll ($r = \boxdot, \boxdot$, or $\boxdot$) it bids 2-r in the hope that by chance the opponent has the same roll because if it did not, the second player would have a winning response most of the time. On a roll of $r = \boxdot$ or $r = \boxdot$ it always bids 1-r because it wins most of time. Either way, as second player ISMCTS selects responses based on the same reasoning, inducing the first player's roll based on their first bid. This also explains the relative exploitability values in Table 1: the first player plays overly safely and hence is hard to exploit, meanwhile the second player best-responds to an expected pure first player strategy, which makes it highly exploitable. Our exploitability experiments shows a similar skewed first vs. second player effect in II-Goofspiel.

Results for OOS variants on LD(1,1) are shown in Figure 2. Firstly, in head-to-head performance we notice OOS wins 55-60% of games against ISMCTS, results for $\delta \in \{0.1, 0.8, 0.9\}$ seem somewhat consistent across variants and time settings, with varied effects for $\delta > 0.9$. Against the 0.001-equilibrium strategy, $\delta = 0.5, 0.65$ seems the most robust across variants and time settings, with varied effects when $\delta > 0.9$. Exploitability of ISMCTS strategies computed by the multi-match method was 0.88, 0.82, 0.79 at search times of 1, 2, and 25 seconds. There are some clear trends for the exploitability of OOS strategies. First, there is an observed decrease in exploitability as time increases, independent of δ. When δ is varied at the same time setting, the exploitability of the global strategy increases with δ. Then, when $\delta = 1$ every time setting for IST converges to the same highly-exploitable point, illustrating the theoretical problems with full targeting manifesting in practice. Interestingly, this is not the case for PST, where $\delta = 1$ is actually always the best-case scenario. In Liar's Dice, this is sensible because there is no reason to believe that any histories outside the public subgame will effect the decisions being made when in the public subgame. The sudden drop from $\delta = 0.99$ to 1 could be caused by the fact that as δ increases the variance of the importance-sampling corrections of the off-match samples grows; when $\delta = 1$ the sampled counterfactual values may be biased, but have much lower variance. PST is less affected than IST by the targeting and appears to have lower exploitability. For each method and all values of δ: increasing the search time decreases exploitability. The exploitability values from Table 1 also show this trend.

We performed one head-to-head experiment consisting of 500 games of PST($\delta = 0.8$) vs. ISMCTS, in LD(2,2) with 25 seconds of search time per move. PST won 256 games, showing a competitive promise on this much larger game with roughly 352 million information sets.

Algorithm	Time	ϵ_1	ϵ_2	ϵ_σ
ISMCTS	1s	0.235	0.574	0.808
IST	1s	0.337	0.311	0.648
PST	1s	0.203	0.211	0.414
ISMCTS	5s	0.251	0.548	0.799
IST	5s	0.229	0.295	0.524
PST	5s	0.148	0.125	0.273

Table 1: LD(1,1) exploitability using full-stitching, $\delta = 0.9$.

Conclusion

In the paper, we have introduced Online Outcome Sampling, the first Monte Carlo Tree Search algorithm that is guaranteed to produce approximate equilibrium strategies as search time per move is increased in imperfect information games. We showed that in head-to-head performance, OOS is able to compete with ISMCTS while producing strategies with lower exploitability at the same search time in Liar's Dice and II-Goofspiel. We propose two methods for targeting relevant parts of the game based on the current match history, IST and PST. In II-Goofspiel, results are only slightly affected by different targeting probabilities, whereas the effect is stronger in LD(1,1), with PST seeming better overall.

In future work, we hope to investigate more and larger games such as no-limit poker, and the effect of informed playout policies. Ultimately, we want to make practical comparisons to other baseline algorithms such as PIMC (Long et al. 2010), MMCTS (Auger 2011), and IIMC (Furtak and Buro 2013). One interesting question is whether the subgame decomposition ideas of (Burch, Johanson, and Bowling 2014) could be adapted to the online search setting. Finally, using MCRNR (Ponsen, de Jong, and Lanctot 2011) as the base algorithm could provide a balance between exploitability and exploitation against known opponents.

Acknowledgments. This work is partially funded by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project Go4Nature, grant number 612.000.938 and the Czech Science Foundation, grant no. P202/12/2054.

References

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2/3):235–256.

Auger, D. 2011. Multiple tree for partially observable Monte-Carlo tree search. In *Applications of Evolutionary Computation (EvoApplications 2011), Part I*, volume 6624 of *LNCS*, 53–62.

Billings, D.; Davidson, A.; Schauenberg, T.; Burch, N.; Bowling, M.; Holte, R.; Schaeffer, J.; and Szafron, D. 2004. Game tree search with adaptation in stochastic imperfect information games. In *In Computers and Games (CG)*.

Browne, C.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Burch, N.; Johanson, M.; and Bowling, M. 2014. Solving imperfect information games using decomposition. In *28th AAAI Conference on Artificial Intelligence*.

Buro, M.; Long, J.; Furtak, T.; and Sturtevant, N. 2009. Im-

proving state evaluation, inference, and search in trick-based card games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1407–1413.

Ciancarini, P., and Favini, G. 2010. Monte carlo tree search in Kriegspiel. *Artificial Intelligence* 174(11):670–684.

Cowling, P. I.; Powley, E. J.; and Whitehouse, D. 2012. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games* 4(2):120–143.

Frank, I.; Basin, D.; and Matsubara, H. 1998. Finding optimal strategies for imperfect information games. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 500–507.

Furtak, T., and Buro, M. 2013. Recursive Monte Carlo search for imperfect information games. In *IEEE Conference on Computational Intelligence in Games (CIG 2013)*.

Ganzfried, S., and Sandholm, T. 2013. Improving performance in imperfect-information games with large state and action spaces by solving endgames. In *AAAI Workshop on Computer Poker and Incomplete Information*.

Gilpin, A., and Sandholm, T. 2006. A competitive Texas Holdem poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, 1453–1454.

Gilpin, A., and Sandholm, T. 2007. Better automated abstraction techniques for imperfect information games, with application to Texas Holdem poker. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Gilpin, A. 2009. *Algorithms for Abstracting and Solving Imperfect Information Games*. Ph.D. Dissertation, Carnegie Mellon University.

Ginsberg, M. 1996. Partition search. In *Proceedings of the Thirteenth National Conference on Artitificial Intelligence (AAAI)*, 228–233.

Ginsberg, M. 2001. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research* 14:303–358.

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

Johanson, M.; Bard, N.; Burch, N.; and Bowling, M. 2012a. Finding optimal abstract strategies in extensive form games. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI)*, 1371–1379.

Johanson, M.; Bard, N.; Lanctot, M.; Gibson, R.; and Bowling, M. 2012b. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Johanson, M. 2007. Robust strategies and counter-strategies: Building a champion level computer Poker player. Master's thesis, University of Alberta.

Kocsis, L., and Szepesvári, C. 2006. Bandit-based Monte Carlo planning. In *15th European Conference on Machine Learning*, volume 4212 of *LNCS*, 282–293.

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

Lanctot, M.; Gibson, R.; Burch, N.; and Bowling, M. 2012. No-regret learning in extensive-form games with imperfect recall. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML 2012)*.

Levy, D. 1989. *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad*. Ellis Horwood Ltd.

Lisy, V.; Bosansky, B.; and Pechoucek, M. 2012. Anytime algorithms for multi-agent visibility-based pursuit-evasion games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1301–1302.

Long, J.; Sturtevant, N. R.; Buro, M.; and Furtak, T. 2010. Understanding the success of perfect information Monte Carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 134–140.

Neller, T. W., and Hnath, S. 2011. Approximating optimal Dudo play with fixed-strategy iteration counterfactual regret minimization. In *Computers and Games*.

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

Ponsen, M.; de Jong, S.; and Lanctot, M. 2011. Computing approximate Nash equilibria and robust best-responses using sampling. *Journal of Artificial Intelligence Research* 42:575–605.

Rubin, J., and Watson, I. 2010. Computer poker: A review. *Artificial Intelligence* 175(5–6):958–987.

Sandholm, T. 2010. The state of solving large incomplete-information games, and application to poker. *AI Magazine* 31(4):13–32.

Shafiei, M.; Sturtevant, N. R.; and Schaeffer, J. 2009. Comparing UCT versus CFR in simultaneous games. In *IJCAI Workshop on General Game-Playing (GIGA)*, 75–82.

Sheppard, B. 2002. World-championship-caliber scrabble. *Artificial Intelligence* 134:241–275.

Sturtevant, N. R. 2008. An analysis of UCT in multi-player games. *ICGA Journal* 31(4):195–208.

Whitehouse, D.; Cowling, P.; Powley, E.; and Rollason, J. 2013. Integrating Monte Carlo tree search with knowledge-based methods to create engaging play in a commercial mobile game. In *9th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 100–106.

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 2007)*.