Monte Carlo Continual Resolving for Online Strategy Computation in Imperfect Information Games

Michal Šustr
michal.sustr@aic.fel.cvut.cz

Vojtěch Kovařík
vojta.kovarik@gmail.com

Viliam Lisý
viliam.lisy@fel.cvut.cz

Artificial Intelligence Center, FEE
Czech Technical University
Prague, Czech Republic

ABSTRACT

Online game playing algorithms produce high-quality strategies with a fraction of memory and computation required by their offline alternatives. Continual Resolving (CR) is a recent theoretically sound approach to online game playing that has been used to outperform human professionals in poker. However, parts of the algorithm were specific to poker, which enjoys many properties not shared by other imperfect information games. We present a domain-independent formulation of CR applicable to any two-player zero-sum extensive-form games (EFGs). It works with an abstract resolving algorithm, which can be instantiated by various EFG solvers. We further describe and implement its Monte Carlo variant (MCCR) which uses Monte Carlo Counterfactual Regret Minimization (MCCFR) as a resolver. We prove the correctness of CR and show an $O(T^{-1/2})$-dependence of MCCR's exploitability on the computation time. Furthermore, we present an empirical comparison of MCCR with incremental tree building to Online Outcome Sampling and Information-set MCTS on several domains.

KEYWORDS

counterfactual regret minimization; resolving; imperfect information; Monte Carlo; online play; extensive-form games; Nash equilibrium

ACM Reference Format:
Michal Šustr, Vojtěch Kovařík, and Viliam Lisý. 2019. Monte Carlo Continual Resolving for Online Strategy Computation in Imperfect Information Games. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Strategies for playing games can be pre-computed *offline* for all possible situations, or computed *online* only for the situations that occur in a particular match. The advantage of the offline computation are stronger bounds on the quality of the computed strategy. Therefore, it is preferable if we want to solve a game optimally. On the other hand, online algorithms can produce strong strategies with a fraction of memory and time requirements of the offline approaches. Online game playing algorithms have outperformed humans in Chess [14], Go [27], and no-limit Poker [3, 22].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

While online approaches have always been the method of choice for strong play in perfect information games, it is less clear how to apply them in imperfect information games (IIGs). To find the optimal strategy for a specific situation in an IIG, a player has to reason about the unknown parts of the game state. They depend on the (possibly unobservable) actions of the opponent prior to the situation, which in turn depends on what the optimal decisions are for both players in many other parts of the game. This makes the optimal strategies in distinct parts of the game closely interdependent and makes correct reasoning about the current situation difficult without solving the game as a whole.

Existing online game playing algorithms for imperfect information games either do not provide any guarantees on the quality of the strategy they produce [8, 9, 21], or require the existence of a compact heuristic evaluation function and a significant amount of computation to construct it [4, 22]. Moreover, the algorithms that are theoretically sound were developed primarily for Texas hold'em poker, which has a very particular information structure. After the initial cards are dealt, all of the actions and chance outcomes that follow are perfectly observable. Furthermore, since the players' moves alternate, the number of actions taken by each player is always known. None of this holds in general for games that can be represented as two-player zero-sum extensive-form games (EFGs). In a blind chess [8], we may learn we have lost a piece, but not necessarily which of the opponent's pieces took it. In visibility-based pursuit-evasion [25], we may know the opponent remained hidden, but not in which direction she moved. In phantom games [28], we may learn it is our turn to play, but not how many illegal moves has the opponent attempted. Because of these complications, the previous theoretically sound algorithms for imperfect-information games are no longer directly applicable.

The sole exception is Online Outcome Sampling (OOS) [20]. It is theoretically sound, completely domain independent, and it does not use any pre-computed evaluation function. However, it starts all its samples from the beginning of the game, and it has to keep sampling actions that cannot occur in the match anymore. As a result, its memory requirements grow as more and more actions are taken in the match, and the high variance in its importance sampling corrections slows down the convergence.

We revisit the Continual Resolving algorithm (CR) introduced in [22] for poker and show how it can be generalized in a way that can handle the complications of general two-player zero-sum EFGs. Based on this generic algorithm, we introduce Monte Carlo Continual Resolving (MCCR), which combines MCCFR [18] with

incremental construction of the game tree, similarly to OOS, but replaces its targeted sampling scheme by Continual Resolving. This leads to faster sampling since MCCR starts its samples not from the root, but from the current point in the game. It also decreases the memory requirements by not having to maintain statistics about parts of the game no longer relevant to the current match. Furthermore, it allows evaluating continual resolving in various domains, without the need to construct expensive evaluation functions.

We prove that MCCR's exploitability approaches zero with increasing computational resources and verify this property empirically in multiple domains. We present an extensive experimental comparison of MCCR with OOS, Information-set Monte Carlo Tree Search (IS-MCTS) [9] and MCCFR. We show that MCCR's performance heavily depends on its ability to quickly estimate key statistics close to the root, which is good in some domains, but insufficient in others.

2 BACKGROUND

We now describe the standard notation for IIGs and MCCFR.

2.1 Imperfect Information Games

We focus on two-player zero-sum extensive-form games with imperfect information. Based on [24], game G can be described by

- $\mathcal{H}$ – the set of *histories*, representing sequences of actions.
- $\mathcal{Z}$ – the set of terminal histories (those $z \in \mathcal{H}$ which are not a prefix of any other history). We use $g \sqsubset h$ to denote the fact that g is equal to or a prefix of h.
- $\mathcal{A}(h) := \{a \mid ha \in \mathcal{H}\}$ denotes the set of actions available at a *non-terminal history* $h \in \mathcal{H} \setminus \mathcal{Z}$. The term ha refers to a history, i.e. child of history h by playing a.
- $\mathcal{P} : \mathcal{H} \setminus \mathcal{Z} \to \{1, 2, c\}$ is the *player function* partitioning non-terminal histories into $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_c$ depending on which player chooses an action at h. Player c is a special player, called "chance" or "nature".
- *The strategy of chance* is a fixed probability distribution σ_c over actions in chance player's histories.
- The *utility function* $u = (u_1, u_2)$ assigns to each terminal history z the rewards $u_1(z), u_2(z) \in \mathbb{R}$ received by players 1 and 2 upon reaching z. We assume that $u_2 = -u_1$.
- The *information-partition* $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2)$ captures the imperfect information of G. For each player $i \in \{1, 2\}$, $\mathcal{I}_i$ is a partition of $\mathcal{H}_i$. If $g, h \in \mathcal{H}_i$ belong to the same $I \in \mathcal{I}_i$ then i cannot distinguish between them. Actions available at infoset I are the same as in each history h of I, therefore we overload $\mathcal{A}(I) := \mathcal{A}(h)$. We only consider games with *perfect recall*, where the players always remember their past actions and the information sets visited so far.

A *behavioral strategy* $\sigma_i \in \Sigma_i$ of player i assigns to each $I \in \mathcal{I}_i$ a probability distribution $\sigma(I)$ over available actions $a \in \mathcal{A}(I)$. A *strategy profile* (or simply *strategy*) $\sigma = (\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$ consists of strategies of players 1 and 2. For a player $i \in \{1, 2\}$, $-i$ will be used to denote the other two actors $\{1, 2, c\} \setminus \{i\}$ in G (for example $\mathcal{H}_{-1} := \mathcal{H}_2 \cup \mathcal{H}_c$) and opp_i denotes i's opponent ($\text{opp}_1 := 2$).

2.2 Nash Equilibria and Counterfactual Values

The *reach probability* of a history $h \in \mathcal{H}$ under σ is defined as $\pi^\sigma(h) = \pi_1^\sigma(h)\pi_2^\sigma(h)\pi_c^\sigma(h)$, where each $\pi_i^\sigma(h)$ is a product of probabilities of the actions taken by player i between the root and h. The reach probabilities $\pi_i^\sigma(h|g)$ and $\pi^\sigma(h|g)$ conditional on being in some $g \sqsubset h$ are defined analogously, except that the products are only taken over the actions on the path between g and h. Finally, $\pi_{-i}^\sigma(\cdot)$ is defined like $\pi^\sigma(\cdot)$, except that in the product $\pi_1^\sigma(\cdot)\pi_2^\sigma(\cdot)\pi_c^\sigma(\cdot)$ the term $\pi_i^\sigma(\cdot)$ is replaced by 1.

The *expected utility* for player i of a strategy profile σ is $u_i(\sigma) = \sum_{z \in \mathcal{Z}} \pi^\sigma(z) u_i(z)$. The profile σ is an *ϵ-Nash equilibrium* (ϵ-NE) if

$$(\forall i \in \{1, 2\}) \ : \ u_i(\sigma) \geq \max_{\sigma_i' \in \Sigma_i} u_i(\sigma_i', \sigma_{\text{opp}_i}) - \epsilon.$$

A Nash equilibrium (NE) is an ϵ-NE with $\epsilon = 0$. It is a standard result that in two-player zero-sum games, all $\sigma^* \in$ NE have the same $u_i(\sigma^*)$ [24]. The *exploitability* $\text{expl}(\sigma)$ of $\sigma \in \Sigma$ is the average of exploitabilities $\text{expl}_i(\sigma)$, $i \in \{1, 2\}$, where

$$\text{expl}_i(\sigma) := u_i(\sigma^*) - \min_{\sigma'_{\text{opp}_i} \in \Sigma_{\text{opp}_i}} u_i(\sigma_i, \sigma'_{\text{opp}_i}).$$

The expected utility conditional on reaching $h \in \mathcal{H}$ is

$$u_i^\sigma(h) = \sum_{h \sqsubset z \in \mathcal{Z}} \pi^\sigma(z|h) u_i(z).$$

An ingenious variant of this concept is the *counterfactual value* (CFV) of a history, defined as $v_i^\sigma(h) := \pi_{-i}^\sigma(h) u_i^\sigma(h)$, and the counterfactual value of taking an action a at h, defined as $v_i^\sigma(h, a) := \pi_{-i}^\sigma(h) u_i^\sigma(ha)$. We set $v_i^\sigma(I) := \sum_{h \in I} v_i^\sigma(h)$ for $I \in \mathcal{I}_i$ and define $v_i^\sigma(I, a)$ analogously. A strategy $\sigma_2^\star \in \Sigma_2$ is a *counterfactual best response* $\text{CBR}(\sigma_1)$ to $\sigma_1 \in \Sigma_1$ if $v_2^{(\sigma_1, \sigma_2^\star)}(I) = \max_{a \in \mathcal{A}(I)} v_2^{(\sigma_1, \sigma_2^\star)}(I, a)$ holds for each $I \in \mathcal{I}_2$ [7].

2.3 Monte Carlo CFR

For a strategy $\sigma \in \Sigma$, $I \in \mathcal{I}_i$ and $a \in \mathcal{A}(I)$, we set the counterfactual regret for not playing a in I under strategy σ to

$$r_i^\sigma(I, a) := v_i^\sigma(I, a) - v_i^\sigma(I). \tag{1}$$

The Counterfactual Regret minimization (CFR) algorithm [29] generates a consecutive sequence of strategies $\sigma^0, \sigma^1, \ldots, \sigma^T$ in such a way that the *immediate counterfactual regret*

$$\bar{R}_{i,\text{imm}}^t(I) := \max_{a \in \mathcal{A}(I)} \bar{R}_{i,\text{imm}}^t(I, a) := \max_{a \in \mathcal{A}(I)} \frac{1}{t} \sum_{t'=1}^{t} r_i^{\sigma^{t'}}(I, a)$$

is minimized for each $I \in \mathcal{I}_i$, $i \in \{1, 2\}$. It does this by using the Regret Matching update rule [1, 12]:

$$\sigma^{t+1}(I, a) := \frac{\max\{\bar{R}_{i,\text{imm}}^t(I, a), 0\}}{\sum_{a' \in \mathcal{A}(I)} \max\{\bar{R}_{i,\text{imm}}^t(I, a'), 0\}}. \tag{2}$$

Since the overall regret is bounded by the sum of immediate counterfactual regrets [29, Theorem 3], this causes the average strategy $\bar{\sigma}^T$ (defined by (3)) to converge to a NE [18, Theorem 1]:

$$\bar{\sigma}^T(I, a) := \frac{\sum_{t=1}^T \pi_i^{\sigma^t}(I) \sigma^t(I, a)}{\sum_{t=1}^T \pi_i^{\sigma^t}(I)} \qquad (\text{where } I \in \mathcal{I}_i). \tag{3}$$

In other words, by accumulating immediate cf. regrets at each information set from the strategies $\sigma^0, \ldots, \sigma^t$, we can produce new

strategy σ^{t+1}. However only the average strategy is guaranteed to converge to NE with $O(1/\sqrt{T})$ – the individual regret matching strategies can oscillate. The initial strategy σ^0 is uniform, but in fact any strategy will work. If the sum in the denominator of update rule (2) is zero, $\sigma^{t+1}(I,a)$ is set to be also uniform.

The disadvantage of CFR is the costly need to traverse the whole game tree during each iteration. Monte Carlo CFR [18] works similarly, but only samples a small portion of the game tree each iteration. It calculates sampled variants of CFR's variables, each of which is an unbiased estimate of the original [18, Lemma 1]. We use a particular variant of MCCFR called Outcome Sampling (OS) [18]. OS only samples a single terminal history z at each iteration, using the sampling strategy $\sigma^{t,\epsilon} := (1-\epsilon)\sigma^t + \epsilon \cdot \text{rnd}$, where $\epsilon \in (0,1]$ controls the exploration and $\text{rnd}(I,a) := \frac{1}{|\mathcal{A}(I)|}$.

This z is then traversed forward (to compute each player's probability $\pi_i^{\sigma^t}(h)$ of playing to reach each prefix of z) and backward (to compute each player's probability $\pi_i^{\sigma^t}(z|h)$ of playing the remaining actions of the history). During the backward traversal, the sampled counterfactual regrets at each visited $I \in \mathcal{I}$ are computed according to (4) and added to $\tilde{R}^T_{i,\text{imm}}(I)$:

$$\tilde{r}_i^{\sigma^t}(I,a) := \begin{cases} w_I \cdot (\pi^{\sigma^t}(z|ha) - \pi^{\sigma^t}(z|h)) & \text{if } ha \sqsubset z \\ w_I \cdot (0 - \pi^{\sigma^t}(z|h)) & \text{otherwise} \end{cases}, \quad (4)$$

where h denotes the prefix of z which is in I and w_I stands for $\frac{1}{\pi^{\sigma^{t,\epsilon}}(z)} \pi_{-i}^{\sigma^t}(z|h) u_i(z)$ [17].

3 DOMAIN-INDEPENDENT FORMULATION OF CONTINUAL RESOLVING

The only domain for which continual resolving has been previously defined and implemented is poker. Poker has several special properties: a) all information sets have a fixed number of histories of the same length, b) public states have the same size and c) only a single player is active in any public state.

There are several complications that occur in more general EFGs: (1) We might be asked to take several turns within a single public state, for example in phantom games. (2) When we are not the acting player, we might be unsure whether it is the opponent's or chance's turn. (3) Finally, both players might be acting within the same public state, for example a secret chance roll determines whether we get to act or not.

In this section, we present an abstract formulation of continual resolving robust enough to handle the complexities of general EFGs. However, we first need to define the concepts like public tree and resolving gadget more carefully.

3.1 Subgames and the Public Tree

To speak about the information available to player i in histories where he doesn't act, we use *augmented information sets*. For player $i \in \{1,2\}$ and history $h \in \mathcal{H} \setminus \mathcal{Z}$, the i's *observation history* $\vec{O}_i(h)$ in h is the sequence $(I_1, a_1, I_2, a_2, \dots)$ of the information sets visited and actions taken by i on the path to h (incl. $I \ni h$ if $h \in \mathcal{H}_i$). Two histories $g, h \in \mathcal{H} \setminus \mathcal{Z}$ belong to the same *augmented information set* $I \in \mathcal{I}_i^{\text{aug}}$ if $\vec{O}_i(g) = \vec{O}_i(h)$. This is equivalent to the definition from [7], except that our definition makes it clear that $\mathcal{I}_i^{\text{aug}}$ is also defined on $\mathcal{H}_i$ (and coincides there with $\mathcal{I}_i$ because of perfect recall).

Remark 3.1 (Alternatives to $\mathcal{I}^{\text{aug}}$). *$\mathcal{I}^{\text{aug}}$ isn't the only viable way of generalizing information sets. One could alternatively consider some further-unrefineable perfect-recall partition $\mathcal{I}_i^*$ of $\mathcal{H}$ which coincides with $\mathcal{I}_i$ on $\mathcal{H}_i$, and many other variants between the two extremes. We focus only on $\mathcal{I}^{\text{aug}}$, since an in-depth discussion of the general topic would be outside of the scope of this paper.*

We use $\sim$ to denote histories indistinguishable by some player:

$$g \sim h \iff \vec{O}_1(g) = \vec{O}_1(h) \vee \vec{O}_2(g) = \vec{O}_2(h).$$

By $\approx$ we denote the transitive closure of $\sim$. Formally, $g \approx h$ iff

$$(\exists n)\,(\exists h_1, \dots, h_n) : g \sim h_1,\ h_1 \sim h_2,\ \dots,\ h_{n-1} \sim h_n,\ h_n \sim h.$$

If two states do *not* satisfy $g \approx h$, then it is common knowledge that both players can tell them apart.

Definition 3.2 (Public state). *Public partition* is any partition $\mathcal{S}$ of $\mathcal{H} \setminus \mathcal{Z}$ whose elements are closed under $\sim$ and form a tree. An element S of such $\mathcal{S}$ is called a *public state*. The *common knowledge partition* $\mathcal{S}_{\text{ck}}$ is the one consisting of the equivalence classes of $\approx$.

Our definition of $\mathcal{S}$ is a reformulation of the definition of [15] in terms of augmented information sets (which aren't used in [15]). The definition of $\mathcal{S}_{\text{ck}}$ is novel. We endow any $\mathcal{S}$ with the tree structure inherited from $\mathcal{H}$. Clearly, $\mathcal{S}_{\text{ck}}$ is the finest public partition. The concept of a public state is helpful for introducing imperfect-information subgames (which aren't defined in [15]).

Definition 3.3 (Subgame). A *subgame* rooted at a public state S is the set $G(S) := \{h \in \mathcal{H} \mid \exists g \in S : g \sqsubset h\}$.

For comparison, [7] defines a subgame as "a forest of trees, closed under both the descendant relation and membership within $\mathcal{I}_i^{\text{aug}}$ for any player". For any $h \in S \in \mathcal{S}_{\text{ck}}$, the subgame rooted at S is the smallest [7]-subgame containing h. As a result, [7]-subgames are "forests of subgames rooted at common-knowledge public states".

We can see that finer public partitions lead to smaller subgames, which are easier to solve. In this sense, the common-knowledge partition is the "best one". However, finding $\mathcal{S}_{\text{ck}}$ is sometimes non-trivial, which makes the definition of general public states from [15] important. The drawback of this definition is its ambiguity — indeed, it allows for extremes such as grouping the whole $\mathcal{H}$ into a single public state, without giving a practical recipe for arriving at the "intuitively correct" public partition.

3.2 Aggregation and the Upper Frontier

Often, it is useful to aggregate reach probabilities and counterfactual values over (augmented) information sets or public states. In general EFGs, an augmented information set $I \in \mathcal{I}_i^{\text{aug}}$ can be "thick", i.e. it can contain both some $ha \in \mathcal{H}$ and it's parent h. This necessarily happens when we are unsure how many actions were taken by other players between our two successive actions. For such I, we only aggregate over the "upper frontier" $\hat{I} := \{h \in I \mid \nexists g \in I : g \sqsubset h \,\&\, g \neq h\}$ of I [10, 11]: We overload $\pi^\sigma(\cdot)$ as $\pi^\sigma(I) := \sum_{h \in \hat{I}} \pi^\sigma(h)$ and $v_i^\sigma(\cdot)$ as $v_i^\sigma(I) := \sum_{h \in \hat{I}} v_i^\sigma(h)$. We define $\hat{S}$ for $S \in \mathcal{S}$, $\pi_i^\sigma(I)$, $\pi_{-i}^\sigma(I)$ and $v_i^\sigma(I,a)$ analogously. By $\hat{S}(i) := \{I \in \mathcal{I}_i^{\text{aug}} \mid \hat{I} \subseteq \hat{S}\}$ we denote the topmost (augmented) information sets of player i in S.

To the best of our knowledge, the issue of "thick" information sets has only been discussed in the context of non-augmented

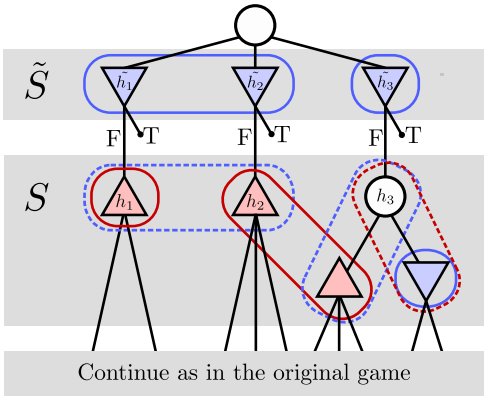

Figure 1: Resolving game $\widetilde{G}(S, \sigma, \tilde{v})$ constructed for player $\triangle$ in a public state S. Player's (augmented) information sets are drawn with solid (resp. dashed) lines of the respective color. The chance node $\bigcirc$ chooses one of $\triangledown$'s histories $\tilde{h}_1, \tilde{h}_2, \tilde{h}_3$, which correspond to the "upper frontier" of S.

information sets in games with imperfect recall [11]. One scenario where thick augmented information sets cause problems is the resolving gadget game, which we discuss next.

3.3 Resolving Gadget Game

We describe a generalization of the resolving gadget game from [7] (cf. [3, 23]) for resolving Player 1's strategy (see Figure 1).

Let $S \in \mathcal{S}$ be a public state to resolve from, $\sigma \in \Sigma$, and let $\tilde{v}(I) \in \mathbb{R}$ for $I \in \hat{S}(i)$ be the required counterfactual values. First, the upper frontier of S is duplicated as $\{\tilde{h} | h \in \hat{S}\} =: \tilde{S}$. Player 2 is the acting player in $\tilde{S}$, and from his point of view, nodes $\tilde{h}$ are partitioned according to $\{\tilde{I} := \{\tilde{h} | h \in \hat{I}\} \mid I \in \hat{S}(2)\}$. In $\tilde{h} \in \tilde{I}$ corresponding to $h \in I$, he can choose between "following" (F) into h and "terminating" (T), which ends the game with utility $\tilde{u}_2(\tilde{h}T) := \tilde{v}(I)\pi^{\sigma}_{-2}(S)/\pi^{\sigma}_{-2}(I)$. From any $h \in \hat{S}$ onward, the game is identical to $G(S)$, except that the utilities are multiplied by a constant: $\tilde{u}_i(z) := u_i(z)\pi^{\sigma}_{-2}(S)$. To turn this into a well-defined game, a root chance node is added and connected to each $h \in \hat{S}$, with transition probabilities $\pi^{\sigma}_{-2}(h)/\pi^{\sigma}_{-2}(S)$.

This game is called the *resolving gadget game* $\widetilde{G}(S, \sigma, \tilde{v})$, or simply $\widetilde{G}(S)$ when there is no risk of confusion, and the variables related to it are denoted by tilde. If $\tilde{\rho} \in \widetilde{\Sigma}$ is a "resolved" strategy in $\widetilde{G}(S)$, we denote the new combined strategy in G as $\sigma^{\text{new}} := \sigma|_{G(S) \leftarrow \tilde{\rho}}$, i.e. play according to strategy $\tilde{\rho}$ in the subgame $G(S)$ and according to σ everywhere else.

The difference between $\widetilde{G}(S, \sigma, \tilde{v})$ and the original version of [7] is that our modification only duplicates the upper frontier $\hat{S}$ and uses normalization constant $\sum_{\hat{S}} \pi^{\sigma}_{-2}(h)$ (rather than $\sum_{S} \pi^{\sigma}_{-2}(h)$) and estimates $\tilde{v}(I) = \sum_{\hat{I}} \tilde{v}(h)$ (rather than $\sum_{I} \tilde{v}(h)$). This enables $\widetilde{G}(S, \sigma, \tilde{v})$ to handle domains with thick information sets and public states. While tedious, it is straightforward to check that $\widetilde{G}(S, \sigma, \tilde{v})$ has all the properties proved in [6, 7, 22]. Without our modification, the resolving games would either sometimes be ill-defined, or wouldn't have the desired properties.

Input : Information set $I \in \mathcal{I}_1$
Output: An action $a \in \mathcal{A}(I)$

1 S $\leftarrow$ the public state which contains I;
2 **if** S $\notin$ KPS **then**
3 $\quad \widetilde{G}(S) \leftarrow$ `BuildResolvingGame(S,D(S))`;
4 $\quad$ KPS $\leftarrow$ KPS $\cup$ S;
5 $\quad$ NPS $\leftarrow$ all $S' \in \mathcal{S}$ where CR acts for the first time after leaving KPS;
6 $\quad \tilde{\rho}, \tilde{\text{D}} \leftarrow$ `Resolve`$(\widetilde{G}(S)$,NPS);
7 $\quad \sigma_1|_{S'} \leftarrow \tilde{\rho}|_{S'}$;
8 $\quad$ D $\leftarrow$ calculate data for NPS based on D, σ_1 and $\tilde{\text{D}}$;
9 **end**
10 **return** a $\sim \sigma_1(I)$

Algorithm 1: Function `Play` of Continual Resolving

The following properties are helpful to get an intuitive understanding of gadget games. Their more general versions and proofs (resp. references for proofs) are listed in the appendix.

Lemma 3.4 (Gadget game preserves opponent's values). *For each $I \in \mathcal{I}_2^{\text{aug}}$ with $I \subset G(S)$, we have $v_2^{\sigma^{\text{new}}}(I) = \tilde{v}_2^{\tilde{\rho}}(I)$.*

Note that the conclusion does *not* hold for counterfactual values of the (resolving) player 1! (This can be easily verified on a simple specific example such as Matching Pennies.)

Lemma 3.5 (Optimal resolving). *If σ and $\tilde{\rho}$ are both Nash equilibria and $\tilde{v}(I) = v_2^{\sigma}(I)$ for each $I \in \hat{S}(2)$, then σ_1^{new} is not exploitable.*

3.4 Continual Resolving

Domain-independent continual resolving closely follows the structure of continual resolving for poker [22], but uses a generalized resolving gadget and handles situations which do not arise in poker, such as multiple moves in one public state. We explain it from the perspective of Player 1. The abstract CR keeps track of strategy σ_1 it has computed in previous moves. Whenever it gets to a public state S, where σ_1 has not been computed, it resolves the subgame $G(S)$. As a by-product of this resolving, it estimates opponent's counterfactual values $v_2^{\sigma_1, \text{CBR}(\sigma_1)}$ for all public states that might come next, allowing it to keep resolving as the game progresses.

CR repetitively calls a `Play` function which takes the current information set $I \in \mathcal{I}_1$ as the input and returns an action $a \in \mathcal{A}(I)$ for Player 1 to play. It maintains the following variables:

- $S \in \mathcal{S}$ ... the current public state,
- KPS $\subset \mathcal{S}$ ... the public states where strategy is known,
- σ_1 ... a strategy defined for every $I \in \mathcal{I}_1$ in KPS,
- NPS $\subset \mathcal{S}$ ... the public states where CR may resolve next,
- $D(S')$ for $S' \in$ NPS ... data allowing resolving at S', such as the estimates of opponent's counterfactual values.

The pseudo-code for CR is described in Algorithm 1. If the current public state belongs to KPS, then the strategy $\sigma_1(I)$ is defined, and we sample action a from it. Otherwise, we should have the data necessary to build some resolving game $\widetilde{G}(S)$ (line 3). We then determine the public states NPS where we might need to resolve next (line 5). We solve $\widetilde{G}(S)$ via some resolving method which also computes the data necessary to resolve from any $S' \in$ NPS (line

6). Finally, we save the resolved strategy in S and update the data needed for resolving (line 7-9). To initialize the variables before the first resolving, we set KPS and σ_1 to $\emptyset$, find appropriate NPS, and start solving the game from the root using the same solver as `Play`, i.e. $\_, D \leftarrow$ `Resolve`(G, NPS).

We now consider CR variants that use the gadget game from Section 3.3 and data of the form $D = (r_1, \tilde{v})$, where $r_1(S') = (\pi_1^{\sigma_1}(h))_{S'}$ is CR's range and $\tilde{v}(S') = (\tilde{v}(J))_J$ estimates opponent's counterfactual value at each $J \in S'(2)$. We shall use the following notation: S_n is the n-th public state from which CR resolves; $\tilde{\rho}_n$ is the corresponding strategy in $\widetilde{G}(S_n)$; σ_1^n is CR's strategy after n-th resolving, defined on KPS_n; the optimal extension of σ_1^n is

$$\sigma_1^{*n} := \text{argmin}_{\nu_1 \in \Sigma_1} \text{expl}_1 \left(\sigma_1^n|_{\text{KPS}_n} \cup \nu_1|_{\mathcal{S} \setminus \text{KPS}_n} \right).$$

Lemmata 24 and 25 of [22] (summarized into Lemma A.5 in our Appendix A) give the following generalization of [22, Theorem S1]:

Theorem 3.6 (Continual resolving bound). *Suppose that CR uses $D = (r_1, \tilde{v})$ and $\widetilde{G}(S, \sigma_1, \tilde{v})$. Then the exploitability of its strategy is bounded by $\text{expl}_1(\sigma_1) \leq \epsilon_0^{\tilde{v}} + \epsilon_1^R + \epsilon_1^{\tilde{v}} + \cdots + \epsilon_{N-1}^{\tilde{v}} + \epsilon_N^R$, where N is the number of resolving steps and $\epsilon_n^R := \widetilde{\text{expl}}_1(\tilde{\rho}_n)$, $\epsilon_n^{\tilde{v}} := \sum_{J \in \hat{S}_{n+1}(2)} \left| \tilde{v}(J) - v_2^{\sigma_1^{*n}, CBR}(J) \right|$ are the exploitability (in $\widetilde{G}(S_n)$) and value estimation error made by the n-th resolver (resp. initialization for $n = 0$).*

The DeepStack algorithm from [22] is a poker-specific instance of the CR variant described in the above paragraph. Its resolver is a modification of CFR with neural network heuristics and sparse look-ahead trees. We make CR domain-independent and allowing for different resolving games (`BuildResolvingGame`), algorithms (`Resolve`), and schemes (by changing line 5).

4 MONTE CARLO CONTINUAL RESOLVING

Monte Carlo Continual Resolving is a specific instance of CR which uses Outcome Sampling MCCFR for game (re)solving. Its data are of the form $D = (r_1, \tilde{v})$ described above and it resolves using the gadget game from Section 3.3. We first present an abstract version of the algorithms that we formally analyze, and then add improvements that make it practical. To simplify the theoretical analysis, we assume MCCFR computes the exact counterfactual value of resulting average strategy $\bar{\sigma}^T$ for further resolving. (We later discuss more realistic alternatives.) The following theorem shows that MCCR's exploitability converges to 0 at the rate of $O(T^{-1/2})$.

Theorem 4.1 (MCCR bound). *With probability at least $(1-p)^{N+1}$, the exploitability of strategy σ computed by MCCR satisfies*

$$\text{expl}_i(\sigma) \leq \left(\sqrt{2}/\sqrt{p} + 1 \right) |\mathcal{I}_i| \frac{\Delta_{u,i}\sqrt{A_i}}{\delta} \left(\frac{2}{\sqrt{T_0}} + \frac{2N-1}{\sqrt{T_R}} \right),$$

where T_0 and T_R are the numbers of MCCR's iterations in pre-play and each resolving, N is the required number of resolvings, $\delta = \min_{z,t} q_t(z)$ where $q_t(z)$ is the probability of sampling $z \in \mathcal{Z}$ at iteration t, $\Delta_{u,i} = \max_{z,z'} |u_i(z) - u_i(z')|$ and $A_i = \max_{I \in \mathcal{I}_i} |\mathcal{A}(I)|$.

The proof is presented in the appendix. Essentially, it inductively combines the OS bound (Lemma A.1) with the guarantees available for resolving games in order to compute the overall exploitability bound.[1] For specific domains, a much tighter bound can be obtained by going through our proof in more detail and noting that the size of subgames decreases exponentially as the game progresses (whereas the proof assumes that it remains constant). In effect, this would replace the N in the bound above by a small constant.

4.1 Practical Modifications

Above, we describe an abstract version of MCCR optimized for clarity and ease of subsequent theoretical analysis. We now describe the version of MCCR that we implemented in practice. The code used for the experiments is available online at https://github.com/aicenter/gtlibrary-java/tree/mccr.

4.1.1 Incremental Tree-Building. A massive reduction in the memory requirements can be achieved by building the game tree incrementally, similarly to Monte Carlo Tree Search (MCTS) [5]. We start with a tree that only contains the root. When an information set is reached that is not in memory, it is added to it and a playout policy (e.g., uniformly random) is used for the remainder of the sample. In playout, information sets are not added to memory. Only the regrets in information sets stored in the memory are updated.

4.1.2 Counterfactual Value Estimation. Since the computation of the exact counterfactual values of the average strategy needed by $\widetilde{G}(S, \sigma, \cdot)$ requires the traversal of the whole game tree, we have to work with their estimates instead. To this end, our MCCFR additionally computes the opponent's *sampled counterfactual values*

$$\tilde{v}_2^{\sigma^t}(I) := \frac{1}{\pi^{\sigma^t,\epsilon}(z)} \pi_{-2}^{\sigma^t}(h) \pi^{\sigma^t}(z|h) u_2(z).$$

It is not possible to compute the exact counterfactual value of the average strategy just from the values of the current strategies. Once the T iterations are complete, the standard way of estimating the counterfactual values of $\bar{\sigma}^T$ is using *arithmetic averages*

$$\tilde{v}(I) := \frac{1}{T} \sum \tilde{v}_2^{\sigma^t}(I). \tag{5}$$

However, we have observed better results with *weighted averages*

$$\tilde{v}(h) := \sum_t \tilde{\pi}^{\sigma^t}(h)\, v_2^{\sigma^t}(h) \;/\; \sum_t \tilde{\pi}^{\sigma^t}(h). \tag{6}$$

The stability and accuracy of these estimates is experimentally evaluated in Section 5 and further analyzed in Appendix B. We also propose an unbiased estimate of the exact values computed from the already executed samples, but its variance makes it impractical.

4.1.3 Root Distribution of Gadgets. As in [22], we use the information about opponent's strategy from previous resolving when constructing the gadget game. Rather than being proportional to $\pi_{-2}(h)$, the root probabilities are proportional to $\pi_{-2}(h)(\pi_2(h) + \epsilon)$. This modification is sound as long as $\epsilon > 0$.

[1] Note that Theorem 4.1 isn't a straightforward corollary of Theorem 3.6, since calculating the numbers $\epsilon_n^{\tilde{v}}$ does require non-trivial work. In particular, $\bar{\sigma}^T$ from the n-th resolving isn't the same as σ_1^{n*}, $CBR(\sigma_1^{n*})$ and the simplifying assumption about $\tilde{v}$ is *not* equivalent to assuming that $\epsilon_n^{\tilde{v}} = 0$.

4.1.4 Custom Sampling Scheme. To improve the efficiency of resolving by MCCFR, we use a custom sampling scheme which differs from OS in two aspects. First, we modify the above sampling scheme such that with probability 90% we sample a history that belongs to the current information set I. This allows us to focus on the most relevant part of the game. Second, whenever $\tilde{h} \in \tilde{S}$ is visited by MCCFR, we sample both actions (T and F). This increases the transparency of the algorithm, since all iterations now "do a similar amount of work" (rather than some terminating immediately). These modifications are theoretically sound, since the resulting sampling scheme still satisfies the assumptions of the general MCCFR bound from [17].

4.1.5 Keeping the Data between Successive Resolvings. Both in pre-play and subsequent resolvings, MCCFR operates on successively smaller and smaller subsets of the game tree. In particular, we don't need to start each resolving from scratch, but we can re-use the previous computations. To do this, we initialize each resolving MCCFR with the MCCFR variables (regrets, average strategy and the corresponding value estimates) from the previous resolving (resp. pre-play). In practice this is accomplished by simply not resetting the data from the previous MCCFR. While not being backed up by theory, this approach worked better in most practical scenarios, and we believe it can be made correct with the use of warm-starting [2] of the resolving gadget.

5 EXPERIMENTAL EVALUATION

After brief introduction of competing methods and explaining the used methodology, we focus on evaluating the alternative methods to estimate the counterfactual values required for resolving during MCCFR. Next, we evaluate how quickly and reliably these values can be estimated in different domains, since these values are crucial for good performance of MCCR. Finally, we compare exploitability and head-to-head performance to competing methods.

5.1 Competing Methods

Information-Set Monte Carlo Tree Search. IS-MCTS [19] runs MCTS samples as in a perfect information game, but computes statistics for the whole information set and not individual states. When initiated from a non-empty match history, it starts samples uniformly from the states in the current information set. We use two selection functions: Upper Confidence bound applied to Trees (UCT) [16] and Regret Matching (RM) [13]. We use the same settings as in [20]: UCT constant 2x the maximal game outcome, and RM with exploration 0.2. In the following, we refer to IS-MCTS with the corresponding selection function by only `UCT` or `RM`.

Online Outcome Sampling. OOS [20] is an online search variant of MCCFR. MCCFR samples from the root of the tree and needs to pre-build the whole game tree. OOS has two primary distinctions from MCCFR: it builds its search tree incrementally and it can bias samples with some probability to any specific parts of the game tree. This is used to target the information sets (OOS-IST) or the public states (OOS-PST) where the players act during a match.

We do not run OOS-PST on domain of IIGS, due to non-trivial biasing of sampling towards current public state.

We further compare to MCCFR with incremental tree building and the random player denoted RND.

5.2 Computing Exploitability

Since the online game playing algorithms do not compute the strategy for the whole game, evaluating exploitability of the strategies they play is more complicated. One approach, called brute-force in [20], suggest ensuring that the online algorithm is executed in each information set in the game and combining the computed strategies. If the thinking time of the algorithm per move is t, it requires $O(t|\mathcal{I}|)$ time to compute one combined strategy and multiple runs are required to achieve statistical significance for randomized algorithms. While this is prohibitively expensive even for the smaller games used in our evaluation, computing the strategy for each public state, instead of each information set is already feasible. We use this approach, however, it means we have to disable the additional targeting of the current information set in the resolving gadget proposed in Section 4.1.4.

There are two options how to deal with the variance in the combined strategies in different runs of the algorithm in order to compute the exploitability of the real strategy realized by the algorithm. The pessimistic option is to compute the exploitability of each combined strategy and average the results. This assumes the opponent knows the random numbers used by the algorithm for sampling in each resolving. A more realistic option is to average the combined strategies from different runs into an *expected strategy* $\bar{\sigma}$ and compute its exploitability. We use the latter.

5.3 Domains

For direct comparison with prior work, we use same domains as in [20] with parametrizations noted in parentheses: Imperfect Information Goofspiel `IIGS(N)`, Liar's Dice `LD(D1,D2,F)` and Generic Poker `GP(T,C,R,B)`. We add Phantom Tic-Tac-Toe `PTTT` to also have a domain with thick public states, and use Biased Rock Paper Scissors `B-RPS` for small experiments. The detailed rules are in Appendix C with the sizes of the domains in Table 2. We use small and large variants of the domains based on their parametrization. Note that large variants are about 10^4 up to 10^{15} times larger than the smaller ones.

5.4 Results

5.4.1 Averaging of Sampled CFVs. As mentioned in Section 4.1.2, computing the exact counterfactual values of the average strategy $\bar{\sigma}^T$ is often prohibitive, and we replace it by using the arithmetic or weighted averages instead. To compare the two approaches, we run MCCFR on the B-RPS domain (which only has a single NE $\sigma*$ to which $\bar{\sigma}^T$ converges) and measure the distance $\Delta v(t)$ between the estimates and the correct action-values $v_1^{\sigma*}(\text{root}, a)$. In Figure 2 (left), the weighted averages converge to the correct values with increasing number of samples, while the arithmetic averages eventually start diverging. The weighted averages are more accurate (see Appendix, Figure 4 for comparison on each domain) and we will use them exclusively in following experiments.

5.4.2 Stability of CFVs. To find a nearly optimal strategy when resolving, MCCR first needs CFVs of such a strategy (Lemma A.5). However, the MCCFR resolver typically won't have enough time to find such $\bar{\sigma}^T$. If MCCR is to work, the CFVs computed by MCCFR

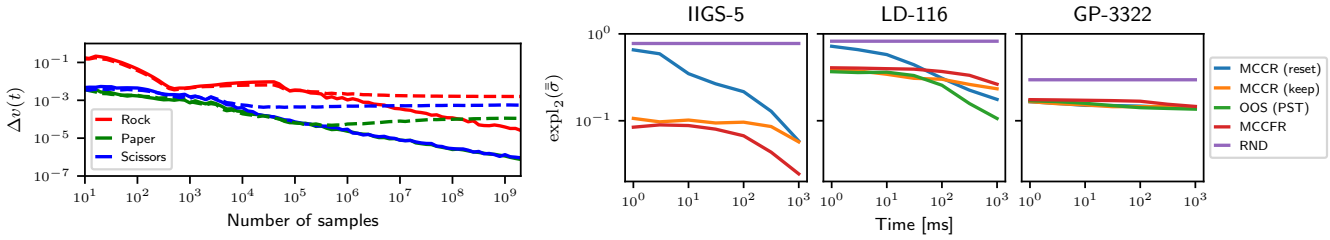

Figure 2: Left – Estimation error of the arithmetic (dashed lines) and weighted averages (solid lines) of action values in B-RPS. Right – Exploitability of $\bar{\bar{\sigma}}$ as a function of the resolving time. All algorithms have pre-play of 300ms.

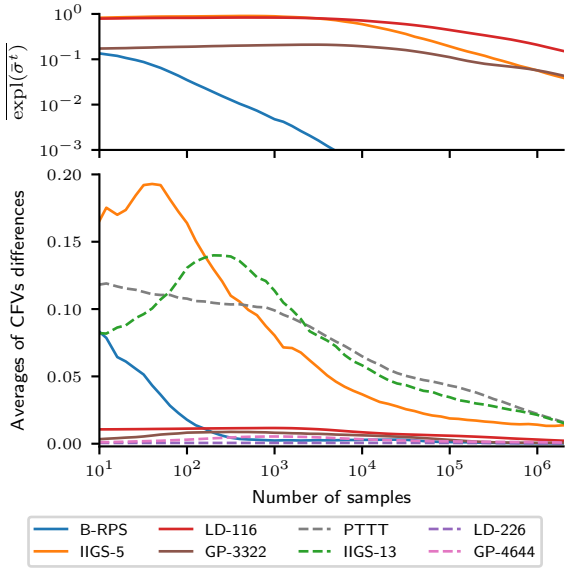

Figure 3: A comparison of exploitability (top) with "CFV instability" (bottom) in different domains. For $t = 10^7$, the differences are 0 by definition.

need to be close to those of an approximate equilibrium CFVs even though they correspond to an exploitable strategy.

We run MCCFR in the root and focus on CFVs in the public states where player 1 acts for the 2nd time (the gadget isn't needed for the first action, and thus neither are CFVs):

$$\Omega := \{J \in \mathcal{I}_2^{\text{aug}} \mid \exists S' \in \mathcal{S}, h \in S' : J \subset S' \,\&\, \text{pl. 1 played once in } h\}.$$

Since there are multiple equilibria MCCFR might converge to, we cannot measure the convergence exactly. Instead, we measure the "instability" of CFV estimates by saving $\tilde{v}_2^t(J)$ for $t \leq T$, tracking how $\Delta_t(J) := |\tilde{v}_2^t(J) - \tilde{v}_2^T(J)|$ changes over time, and aggregating it into $\frac{1}{|\Omega|}\sum_J \Delta_t(J)$. We repeat this experiment with 100 different MCCFR seeds and measure the expectation of the aggregates and, for the small domains, the expected exploitability of $\bar{\sigma}^t$. If the resulting "instability" is close to zero after 10^5 samples (our typical time budget), we consider CFVs to be sufficiently stable.

Figure 3 confirms that in small domains (LD, GP), CFVs stabilize long before the exploitability of the strategy gets low. The error still decreases in larger games (GS, PTTT), but at a slow rate.

5.4.3 Comparison of Exploitability with Other Algorithms. We compare the exploitability of MCCR to OOS-PST and MCCFR, and include random player for reference. We do not include OOS-IST, whose exploitability is comparable to that of OOS-PST [20]. For an evaluation of IS-MCTS's exploitability (which is very high, with the exception of poker) we refer the reader to [19, 20].

Figure 2 (right) confirms that for all algorithms, the exploitability decreases with the increased time per move. MCCR is better than MCCFR on LD and worse on IIGS. The keep variant of MCCR is initially less exploitable than the reset variant, but improves slower. This suggests the keep variant could be improved.

5.4.4 Influence of the Exploration Rate. One of MCCR's parameters is the exploration rate ϵ of its MCCFR resolver. When measuring the exploitability of MCCR we observed no noteworthy influence of ϵ (for $\epsilon = 0.2, 0.4, 0.6, 0.8$, across all of the evaluated domains).

5.4.5 Head-to-head Performance. For each pair of algorithms, thousands of matches have been played on each domain, alternating positions of player 1 and 2. In the smaller (larger) domains, players have 0.3*s* (5s) of pre-play computation and 0.1*s* (1s) per move. Table 1 summarizes the results as percentages of the maximum domain payoff.

Note that the results of the matches are not necessarily transitive, since they are not representative of the algorithms' exploitability. When computationally tractable, the previous experiment 5.4.3 is therefore a much better comparison of an algorithm's performance.

Both variants of MCCR significantly outperform the random opponent in all games. With the exception of PTTT, they are also at least as good as the MCCFR "baseline". This is because public states in PTTT represent the number of moves made, which results in a non-branching public tree and resolving games occupy the entire level as in the original game. MCCR is better than OOS-PST in LD and GP, and better than OOS-IST in large IIGS. MCCR is worse than IS-MCTS on all games with the exception of small LD. However, this does not necessarily mean that MCCR's exploitability is higher than for IS-MCTS in the larger domains, MCCR only fails to find the strategy that would exploit IS-MCTS.

6 CONCLUSION

We propose a generalization of Continual Resolving from poker [22] to other extensive-form games. We show that the structure of the public tree may be more complex in general, and propose an extended version of the resolving gadget necessary to handle this

IIGS-5	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	0.4 ± 1.2	0.6 ± 1.3	–	-2.6 ± 1.3	-2.8 ± 1.3	-6.5 ± 1.2	51.2 ± 1.2
MCCR (reset)		-0.8 ± 1.2	–	-2.5 ± 1.2	-5.5 ± 1.2	-8.1 ± 1.2	49.1 ± 1.2
MCCFR			–	-1.3 ± 1.3	-2.7 ± 1.3	-5.6 ± 1.2	48.5 ± 1.2
OOS-PST				–	–	–	–
OOS-IST					-2.0 ± 1.3	-5.2 ± 1.2	54.5 ± 1.1
RM						-16.6 ± 1.2	67.8 ± 1.0
UCT							70.0 ± 1.0

IIGS-13	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	-18.8 ± 5.6	12.8 ± 5.7	–	-7.3 ± 5.7	-56.4 ± 4.7	-69.1 ± 4.1	43.9 ± 5.1
MCCR (reset)		24.4 ± 5.5	–	4.9 ± 2.6	-35.6 ± 5.3	-56.0 ± 4.7	55.6 ± 4.7
MCCFR			–	-22.8 ± 5.6	-59.9 ± 4.6	-75.1 ± 3.7	37.8 ± 5.3
OOS-PST				–	–	–	–
OOS-IST					-44.4 ± 5.1	-61.2 ± 4.5	58.2 ± 4.6
RM						-22.8 ± 5.6	82.3 ± 3.2
UCT							91.2 ± 2.3

LD-116	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	-1.4 ± 2.0	9.7 ± 2.0	5.2 ± 2.0	-4.1 ± 2.0	2.6 ± 1.0	5.6 ± 2.0	60.9 ± 1.6
MCCR (reset)		11.6 ± 2.0	10.1 ± 2.0	-3.9 ± 2.0	-5.5 ± 2.0	5.3 ± 2.0	60.5 ± 1.6
MCCFR			-6.8 ± 2.0	-8.7 ± 2.0	-4.7 ± 2.0	1.9 ± 1.0	54.0 ± 1.7
OOS-PST				-4.3 ± 2.0	-3.0 ± 2.0	6.5 ± 2.0	60.3 ± 1.6
OOS-IST					-1.7 ± 1.0	5.2 ± 2.0	64.8 ± 1.6
RM						5.2 ± 2.0	66.1 ± 1.5
UCT							65.4 ± 1.5

LD-226	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	6.2 ± 5.4	45.3 ± 5.7	46.1 ± 5.7	-22.7 ± 5.9	-33.6 ± 5.7	-33.4 ± 5.7	76.2 ± 4.2
MCCR (reset)		37.8 ± 5.4	44.2 ± 5.7	-31.2 ± 5.7	-39.8 ± 5.4	-44.8 ± 5.2	81.6 ± 4.5
MCCFR			-5.4 ± 4.6	-55.7 ± 4.5	-49.3 ± 4.6	-47.8 ± 5.1	45.8 ± 5.2
OOS-PST				-53.6 ± 5.3	-51.5 ± 4.9	-46.4 ± 5.1	49.6 ± 4.1
OOS-IST					-12.0 ± 5.6	-22.5 ± 5.6	83.8 ± 3.8
RM						-11.6 ± 5.7	79.7 ± 3.5
UCT							75.4 ± 3.8

GP-3322	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	0.2 ± 0.4	2.2 ± 0.5	1.7 ± 0.5	0.4 ± 0.2	-0.5 ± 0.4	-1.5 ± 0.4	5.9 ± 0.5
MCCR (reset)		0.7 ± 0.4	0.4 ± 0.2	-0.3 ± 0.2	-0.5 ± 0.4	-1.0 ± 0.3	5.5 ± 0.4
MCCFR			-1.9 ± 0.6	-2.7 ± 0.6	-3.6 ± 0.5	-3.3 ± 0.5	6.0 ± 0.6
OOS-PST				-1.0 ± 0.9	-2.1 ± 0.5	-2.8 ± 0.4	7.4 ± 0.6
OOS-IST					-1.3 ± 0.5	-2.1 ± 0.4	7.4 ± 0.6
RM						-1.2 ± 0.4	7.8 ± 0.5
UCT							6.3 ± 0.4

GP-4644	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	1.6 ± 1.2	7.3 ± 2.6	14.2 ± 2.5	-3.4 ± 2.3	-4.1 ± 1.8	-6.9 ± 1.5	19.3 ± 2.6
MCCR (reset)		9.5 ± 2.0	11.9 ± 2.0	-3.5 ± 1.8	-3.0 ± 1.5	-2.5 ± 1.3	15.8 ± 2.2
MCCFR			-8.7 ± 3.1	-13.3 ± 2.9	-9.6 ± 2.3	-6.8 ± 1.9	12.0 ± 3.0
OOS-PST				-8.1 ± 3.0	-8.9 ± 2.4	-5.0 ± 2.0	11.8 ± 3.1
OOS-IST					-2.1 ± 1.8	-1.6 ± 1.2	20.4 ± 2.9
RM						-0.3 ± 1.1	20.5 ± 2.3
UCT							17.6 ± 2.0

PTTT	MCCR (reset)	MCCFR	OOS-PST	OOS-IST	RM	UCT	RND
MCCR (keep)	17.7 ± 3.8	-1.1 ± 0.9	-1.8 ± 1.6	-5.0 ± 3.7	-6.9 ± 3.7	-6.2 ± 3.7	25.5 ± 3.8
MCCR (reset)		-6.2 ± 3.8	-9.8 ± 3.7	-14.6 ± 3.6	-20.9 ± 3.6	-14.3 ± 3.7	21.6 ± 3.7
MCCFR			0.1 ± 3.7	-2.1 ± 1.5	-5.2 ± 3.7	-4.0 ± 3.6	27.9 ± 3.7
OOS-PST				-5.6 ± 3.7	-5.7 ± 3.7	-5.2 ± 3.7	29.4 ± 3.7
OOS-IST					-3.5 ± 3.2	-3.9 ± 3.6	35.1 ± 3.6
RM						5.6 ± 3.6	51.3 ± 3.3
UCT							52.6 ± 3.3

Table 1: Head-to-head performance. Positive numbers mean that the row algorithm is winning against the column algorithm by the given percentage of the maximum payoff in the domain. Gray numbers indicate the winner isn't statistically significant.

complexity. Furthermore, both players may play in the same public state (possibly multiple times), and we extend the definition of Continual Resolving to allow this case as well. We present a completely domain-independent version of the algorithm that can be applied to any EFG, is sufficiently robust to use variable resolving schemes, and can be coupled with different resolving games and algorithms (including classical CFR, depth-limited search utilizing neural networks, or other domain-specific heuristics). We show that the existing theory naturally translates to this generalized case.

We further introduce Monte Carlo CR as a specific instance of this abstract algorithm that uses MCCFR as a resolver. It allows deploying continual resolving on any domain, without the need for expensive construction of evaluation functions. MCCR is theoretically sound as demonstrated by Theorem 4.1, constitutes an improvement over MCCFR in the online setting in terms head-to-head performance, and doesn't have the restrictive memory requirements of OOS. The experimental evaluation shows that MCCR is very sensitive to the quality of its counterfactual value estimates. With good estimates, its worst-case performance (i.e. exploitability) improves faster than that of OOS. In head-to-head matches MCCR plays similarly to OOS, but it only outperforms IS-MCTS in one of the smaller tested domains. Note however that the lack of theoretical guarantees of IS-MCTS often translates into severe exploitability in practice [20], and this cannot be alleviated by increasing IS-MCTS's computational resources [19]. In domains where MCCR's counterfactual value estimates are less precise, its exploitability still converges to zero, but at a slower rate than OOS, and its head-to-head performance is noticeably weaker than that of both OOS and IS-MCTS.

In the future work, the quality of MCCR's estimates might be improved by variance reduction [26], exploring ways of improving these estimates over the course of the game, or by finding an alternative source from which they can be obtained. We also plan to test the hypothesis that there are classes of games where MCCR performs much better than the competing algorithms (in particular, we suspect this might be true for small variants of turn-based computer games such as Heroes of Might & Magic or Civilization).

ACKNOWLEDGMENTS

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the program "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated. This work was supported by Czech science foundation grant no. 18-27483Y.

REFERENCES

[1] David Blackwell and others. 1956. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.* 6, 1 (1956), 1–8.

[2] Noam Brown and Tuomas Sandholm. 2016. Strategy-Based Warm Starting for Regret Minimization in Games.. In *AAAI*. 432–438.

[3] Noam Brown and Tuomas Sandholm. 2017. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems*. 689–699.

[4] Noam Brown, Tuomas Sandholm, and Brandon Amos. 2018. Depth-Limited Solving for Imperfect-Information Games. *arXiv preprint arXiv:1805.08195* (2018).

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[6] Neil Burch. 2017. *Time and Space: Why Imperfect Information Games are Hard*. Ph.D. Dissertation. University of Alberta.

[7] Neil Burch, Michael Johanson, and Michael Bowling. 2014. Solving Imperfect Information Games Using Decomposition.. In *AAAI*. 602–608.

[8] Paolo Ciancarini and Gian Piero Favini. 2010. Monte Carlo tree search in Kriegspiel. *Artificial Intelligence* 174 (July 2010), 670–684. Issue 11.

[9] Peter I Cowling, Edward J Powley, and Daniel Whitehouse. 2012. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 2 (2012), 120–143.

[10] Joseph Y Halpern. 1997. On ambiguities in the interpretation of game trees. *Games and Economic Behavior* 20, 1 (1997), 66–96.

[11] Joseph Y Halpern and Rafael Pass. 2016. Sequential Equilibrium in Games of Imperfect Recall.. In *KR*. 278–287.

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

[13] Sergiu Hart and Andreu Mas-Colell. 2001. A reinforcement procedure leading to correlated equilibrium. In *Economics Essays*. Springer, 181–200.

[14] Feng-Hsiung Hsu. 2006. *Behind Deep Blue: Building the Computer that Defeated the World Chess Championship*. Princeton University Press.

[15] Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. 2011. Accelerating best response calculation in large extensive games. In *IJCAI*, Vol. 11. 258–265.

[16] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[17] Marc Lanctot. 2013. *Monte Carlo sampling and regret minimization for equilibrium computation and decision-making in large extensive form games*. Ph.D. Dissertation. University of Alberta.

[18] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. 2009. Monte Carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*. 1078–1086.

[19] Viliam Lisy. 2014. Alternative selection functions for Information Set Monte Carlo tree search. *Acta Polytechnica* 54, 5 (2014), 333–340.

[20] Viliam Lisý, Marc Lanctot, and Michael Bowling. 2015. Online monte carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 27–36.

[21] Jeffrey Long, Nathan R Sturtevant, Michael Buro, and Timothy Furtak. 2010. Understanding the success of perfect information monte carlo sampling in game tree search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 134–140.

[22] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356, 6337 (2017), 508–513.

[23] Matej Moravcik, Martin Schmid, Karel Ha, Milan Hladik, and Stephen J Gaukrodger. 2016. Refining Subgames in Large Imperfect Information Games.. In *AAAI*. 572–578.

[24] Martin J Osborne and Ariel Rubinstein. 1994. *A course in game theory*. MIT press.

[25] Eric Raboin, Dana Nau, Ugur Kuter, Satyandra K Gupta, and Petr Svec. 2010. Strategy generation in multi-agent imperfect-information pursuit games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 947–954.

[26] Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. 2018. Variance Reduction in Monte Carlo Counterfactual Regret Minimization (VR-MCCFR) for Extensive Form Games using Baselines. *arXiv preprint arXiv:1809.03057* (2018).

[27] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[28] Fabien Teytaud and Olivier Teytaud. 2011. Lemmas on partial observation, with application to phantom games. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*. IEEE, 243–249.

[29] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*. 1729–1736.