Sound Algorithms in Imperfect Information Games

Michal Šustr,[1,2] Martin Schmid,[2] Matej Moravčík,[2] Neil Burch,[2] Marc Lanctot,[2] Michael Bowling[2]
[1] Czech Technical University, [2] DeepMind
Corresponding authors: michal.sustr@aic.fel.cvut.cz, mschmid@google.com

Abstract

Search has played a fundamental role in computer game research since the very beginning. And while online search has been commonly used in perfect information games such as Chess and Go, online search methods for imperfect information games have only been introduced relatively recently. This paper addresses the question of what is a sound online algorithm in an imperfect information setting of two-player zero-sum games. We argue that the fixed-strategy definitions of exploitability and ϵ-Nash equilibria are ill-suited to measure an online algorithm's worst-case performance. We thus formalize ϵ-soundness, a concept that connects the worst-case performance of an online algorithm to the performance of an ϵ-Nash equilibrium. As ϵ-soundness can be difficult to compute in general, we introduce a consistency framework — a hierarchy that connects an online algorithm's behavior to a Nash equilibrium. These multiple levels of consistency describe in what sense an online algorithm plays "just like a fixed Nash equilibrium". These notions further illustrate the difference between perfect and imperfect information settings, as the same consistency guarantees have different worst-case online performance in perfect and imperfect information games. The definitions of soundness and the consistency hierarchy finally provide appropriate tools to analyze online algorithms in repeated imperfect information games. We thus inspect some of the previous online algorithms in a new light, bringing new insights into their worst-case performance guarantees.

1 Introduction

From the very dawn of computer game research, search was a fundamental component of many algorithms. Turing's chess algorithm from 1950 was able to think two moves ahead (Copeland 2004), and Shannon's work on chess from 1950 includes an extensive section on how an evaluation function can be used within search (Shannon 1950). Samuel's checkers algorithm from 1959 already combines search and learning of a value function, approximated through a self-play method and bootstrapping (Samuel 1959). The combination of search and learning has been a crucial component in the remarkable milestones where computers outperformed their human counterparts in challenging games: DeepBlue in Chess (Campbell, Hoane Jr, and Hsu 2002), AlphaGo in Go (Silver et al. 2017), DeepStack and Libratus in Poker (Moravcik et al. 2017; Brown and Sandholm 2018).

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Online methods for approximating Nash equilibria in sequential imperfect information games appeared only in the last few years (Lisý, Lanctot, and Bowling 2015; Brown and Sandholm 2017; Moravcik et al. 2017; Brown and Sandholm 2018, 2019; Brown et al. 2020). We thus investigate what it takes for an online algorithm to be sound in imperfect information settings. While it has been known that search with imperfect information is more challenging than with perfect information (Frank and Basin 1998; Lisý, Lanctot, and Bowling 2015), the problem is more complex than previously thought. Online algorithms "live" in a fundamentally different setting, and they need to be evaluated appropriately.

Previously, a common approach to evaluate online algorithms was to compute a corresponding offline strategy by "querying" the online algorithm at each state ("tabularization" of the strategy) (Lisý, Lanctot, and Bowling 2015; Šustr, Kovařík, and Lisý 2019). One would then report the exploitability of the resulting offline strategy. We show that this is not generally possible and that naive tabularization can also lead to incorrect conclusions about the online algorithm's worst-case performance. As a consequence we show that some algorithms previously considered to be sound are not.

We first give a simple example of how an online algorithm can lose to an adversary in a repeated game setting. Previously, such an algorithm would be considered optimal based on a naive tabularization. We build on top of this example to introduce a framework for properly evaluating an online algorithm's performance. Within this framework, we introduce the definition of a sound and ϵ-sound algorithm. Like the exploitability of a strategy in the offline setting, the soundness of an algorithm is a measure of its performance against a worst-case adversary. Importantly, this notion collapses to the previous notion of exploitability when the algorithm follows a fixed strategy profile.

We then introduce a consistency framework, a hierarchy that formally states in what sense an online algorithm plays "consistently" with an ϵ-equilibrium. The hierarchy allows stating multiple bounds on the algorithm's soundness, based on the ϵ-equilibrium and consistency type. The stronger the consistency is in our hierarchy, the stronger are the bounds. This further illustrates the discrepancy of search in perfect

and imperfect information settings, as these bounds sometimes differ for perfect and imperfect information games.

The definitions of soundness and the consistency hierarchy finally provide appropriate tools to analyze online algorithms in imperfect information games. We thus inspect some of the previous online algorithms in a new light, bringing new insights into their worst-case performance guarantees. Namely, we focus on the Online Outcome Sampling (OOS) (Lisý, Lanctot, and Bowling 2015) algorithm. Consider the following statement from the OOS publication: "We show that OOS is consistent, i.e., it is guaranteed to converge to an equilibrium strategy as search time increases. To the best of our knowledge, this is not the case for any existing online game playing algorithm. . . ' The problem is that OOS provides only the weakest of the introduced consistencies — local consistency. As the local consistency gives no guarantee for imperfect information games (in contrast to perfect information games), OOS (and potentially other locally consistent algorithms) can be highly exploited by an adversary. The experimental section then confirms this issue for OOS in two small imperfect information games.

2 Background

We present our results using the recent formalism of factored-observations games (Kovařík et al. 2019). The entirety of the paper trivially applies to the extensive form formalism (Osborne and Rubinstein 1994) as well[1] (as we are only relying on the notion of states and rewards). We believe this choice of formalism makes it easier to incorporate our definitions in the future online algorithms, as sound search in imperfect information critically relies on the notion of common/public information (Burch, Johanson, and Bowling 2014; Seitz et al. 2019). Indeed, all the recently introduced online algorithms in imperfect information games rely on these notions (Moravcik et al. 2017; Brown and Sandholm 2018; Šustr, Kovařík, and Lisý 2019).

Definition 1. *A factored-observations game is a tuple*

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{W}, w^o, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O} \rangle,$$

where:

- $\mathcal{N} = \{1, 2\}$ *is a* ***player set****. We use symbol n for a player and -n for its opponent.*
- $\mathcal{W}$ *is a set of* ***world states*** *and $w^0 \in \mathcal{W}$ is a designated initial world state.*
- $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ *is a space of* ***joint actions****. The subsets $\mathcal{A}_n(w) \subset \mathcal{A}_n$ and $\mathcal{A}(w) = \mathcal{A}_1(w) \times \mathcal{A}_2(w) \subset \mathcal{A}$ specify the (joint) actions legal at $w \in \mathcal{W}$. For $a \in \mathcal{A}$, we write $a = (a_1, a_2)$. $\mathcal{A}_n(w)$ for $n \in \mathcal{N}$ are either all non-empty or all empty. A world state with no legal actions is* ***terminal****.*
- *After taking a (legal) joint action a at w, the* ***transition function*** *$\mathcal{T}$ determines the next world state w', drawn from the probability distribution $\mathcal{T}(w, a) \in \Delta(\mathcal{W})$.*
- $\mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2)$*, and $\mathcal{R}_n(w, a)$ is the* ***reward*** *player n receives when a joint action a is taken at w.*
- $\mathcal{O} = (\mathcal{O}_{\text{priv(1)}}, \mathcal{O}_{\text{priv(2)}}, \mathcal{O}_{\text{pub}})$ *is the* ***observation function****, where $\mathcal{O}_{(\cdot)} : \mathcal{W} \times \mathcal{A} \times \mathcal{W} \to \mathbb{O}_{(\cdot)}$ specifies the* ***private observation*** *that player n receives, resp. the* ***public observation*** *that every player receives, upon transitioning from world state w to w' via some joint action a.*

A legal **world history** (or trajectory) is a finite sequence $h = (w^0, a^0, w^1, a^1, \ldots, w^t)$, where $w^k \in \mathcal{W}$, $a^k \in \mathcal{A}(w^k)$, and $w^{k+1} \in \mathcal{W}$ is in the support of $\mathcal{T}(w^k, a^k)$. We denote the set of all legal histories by $\mathcal{H}$, and the set of all sub-sequences of h that are legal histories as $\mathcal{H}(h) \subseteq \mathcal{H}$.

Since the last world state in each $h \in \mathcal{H}$ is uniquely defined, the notation for $\mathcal{W}$ can be overloaded to work with $\mathcal{H}$ (e.g., $\mathcal{A}(h) := \mathcal{A}(\text{the last } w \text{ in } h)$, h being terminal, ...). We use $\mathcal{Z}$ to denote the set of all terminal histories, i.e. histories where the last world state is terminal.

The **cumulative reward** of n at h is $\sum_{k=0}^{t-1} r_n^k := \sum_{k=0}^{t-1} \mathcal{R}_n(w^k, a^k)$. When h is a terminal history, cumulative rewards can also be called **utilities**, and denoted as $u_n(z)$. We assume games are **zero-sum**, so $u_n(z) = -u_{\text{-}n}(z)\ \forall z \in \mathcal{Z}$. The maximum difference of utilities is $\Delta = |\max_{z \in \mathcal{Z}} u_1(z) - \min_{z \in \mathcal{Z}} u_1(z)|$

Player n's **information state** or **private history** at $h = (w^0, a^0,$
$w^1, a^1, \ldots, w^t)$ is the action-observation sequence $s_n(h) := (O_n^0, a_n^0,$
$O_n^1, a_n^1, \ldots, O_n^t)$, where $O_n^k = \mathcal{O}_n(w^{k-1}, a^{k-1}, w^k)$ and O_n^0 is some initial observation. The space $\mathcal{S}_n$ of all such sequences can be viewed as the **private tree** of n.

A **strategy profile** is a pair $\sigma = (\sigma_1, \sigma_2)$, where each (behavioral) **strategy** $\sigma_n : s_n \in \mathcal{S}_n \mapsto \sigma_n(s_n) \in \Delta(\mathcal{A}_n(s_n))$ specifies the probability distribution from which player n draws their next action (conditional on having information s_n). We denote the set of all strategies of player n as Σ_n and the set of all strategy profiles as Σ.

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 $\pi_n^\sigma(h)$ is a product of probabilities of the actions taken by player n between the root and h, and $\pi_c^\sigma(h)$ is the product of stochastic transitions. The **expected utility** for player n of a strategy profile σ is $u_n(\sigma) = \sum_{z \in \mathcal{Z}} \pi^\sigma(z) u_n(z)$.

We define a **best response** of player n to the other player's strategies $\sigma_{\text{-}n}$ as a strategy $br(\sigma_{\text{-}n}) \in \arg\max_{\sigma'_n \in \Sigma_n} u_n(\sigma'_n, \sigma_{\text{-}n})$ and **best response value** $brv(\sigma_{\text{-}n}) = \max_{\sigma'_n \in \Sigma_n} u_n(\sigma'_n, \sigma_{\text{-}n})$. The profile σ is an **ϵ-Nash equilibrium** if $(\forall n \in \mathcal{N}) : u_n(\sigma) \geq \max_{\sigma'_n \in \Sigma_n} u_n(\sigma'_n,$
$\sigma_{\text{-}n}) - \epsilon$, and we denote the set of all ϵ-equilibrium strategies of player n as $\mathcal{NE}_n^\epsilon$. The **strategy exploitability** is $expl_n(\sigma_n) := \left[u_n(\sigma^*) - \min_{\sigma'_{\text{-}n} \in \Sigma_{\text{-}n}} u_n(\sigma_n, \sigma'_{\text{-}n})\right]$ where σ^* is an equilibrium strategy. The **game value** $u^* = u_1(\sigma^*)$ is the utility player 1 can achieve under a Nash equilibrium strategy profile.

3 Online Algorithm

The environment we are concerned with is that of a repeated game, consisting of a sequence of individual matches. As a match progresses, the algorithm produces a strategy for a

[1] Under the assumption the games are perfect-recall and 1-timeable (Kovařík et al. 2019).

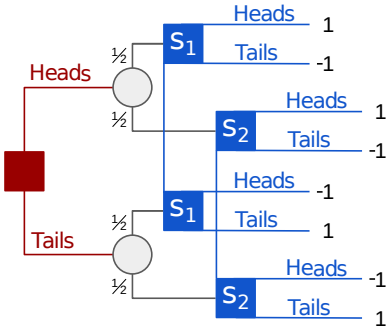

Figure 1: Coordinated Matching Pennies. After the first player acts, chance randomly chooses whether the second player will be playing in the information state s_1 or s_2. The first player receives utility of 1 if players' actions match and -1 if they mismatch.

visited information state on-line: that is, once it actually observes the state. This common framework of repeated games is particularly suitable for analysis of online algorithms, as the online algorithm can be conditioned on the past experience (e.g. by trying to adapt to the opponent or by re-using parts of the previous computation). We are then interested in the accumulated reward of the agent during the span of the repeated game. Of particular interest will be the expected reward against a worst-case adversary.

Coordinated Matching Pennies

We now introduce a small imperfect information game that will be used throughout the article – "Coordinated Matching Pennies" (CMP). It is a variation on the well-known Matching Pennies game (Osborne and Rubinstein 1994), where players choose either Heads or Tails and receive a utility of ± 1 if their actions (mis)match. For CMP, we additionally introduce a publicly observable chance event just after the first player acts. See Figure 1 for details.

Let p and q denote the probability of playing Heads in information states s_1 and s_2 respectively. An equilibrium strategy for the second player (Blue) is then any strategy where the average of p and q is $\frac{1}{2}$. He thus has to coordinate the actions between his two information states, while the first player has a unique uniform equilibrium strategy. Similar equilibrium coordination happens also in Kuhn Poker (Kuhn 1950).

Naive Tabularization

We now show that if one naively tries to convert an online algorithm into a fixed strategy, the resulting exploitability is not always representative of the worst-case performance of the online algorithm. Consider the following algorithm `PlayCache` for the repeated game of CMP. `PlayCache` keeps an internal state, a cache – a mapping of information state to probability distribution over the actions, and it gradually fills the cache during the game.

Concretely, `PlayCache` plays for the second player as follows:

- Initialize algorithm's state $\boldsymbol{\theta}_0$ to an empty cache.
- Given an information state s visited during a game, there are three possible cases: i) The cache is empty: play Heads and store $\{s, \text{Heads}\}$ into the cache. ii) The cache is non-empty and contains s: play the cached strategy for s. iii) The cache is non-empty and does not contain s: play Tails and store $\{s, \text{Tails}\}$.

Consider what happens if one tries to naively tabularize the `PlayCache` by querying the algorithm for all the information states. If we query the algorithm for states s_1, s_2, we get the resulting offline strategy s_1 : Heads, s_2 : Tails. Querying the algorithm for states in reverse order, i.e. s_2, s_1 results in s_1 : Tails, s_2 : Heads. And while both of these offline strategies have zero exploitability, one can still exploit the algorithm during the repeated game. This follows from the fact that the very first time the `PlayCache` gets to act, it always plays Heads. The first player can thus simply play Heads during the first match and is guaranteed to win the match. As we will show later, `PlayCache` falls within a class of algorithms that can be exploited, but where the average reward is guaranteed to converge to the game value as we repeatedly keep playing the game.

Where did this discrepancy between the exploitability of the tabularized strategy and the exploitability of the online algorithm come from? It is simply because the tabularized strategy does not properly describe the game dynamics of `PlayCache`. In fact, there is no fixed strategy that does so! We will now formalize an appropriate framework to describe the rewards and dynamics of online algorithms, which will allow us to define notions for the expected reward and the worst-case performance in the online setting.

Online Settings

The **repeated game** p consists of a finite sequence of k individual matches $m = (z_1, z_2, \ldots, z_k)$, where each **match** $z_i \in \mathcal{Z}$ is a sequence of world states and actions $z_i = (w_i^0, a_i^0, w_i^1, a_i^1 \ldots, a_i^{l_i-1}, w_i^{l_i})$, ending in a terminal world state $w_i^{l_i}$. For each visited world state in the match, there is a corresponding information state, i.e. a player's private perspective of the game (for perfect information games, the notion of information state and world state collapse as the player gets to observe the world perfectly). An online algorithm Ω then simply maps an information state observed during a match to a strategy, while possibly using its internal algorithm state (Def. 2).

Given two players that use algorithms Ω_1, Ω_2, we use $P^k_{\Omega_1,\Omega_2}$ to denote the distribution over all the possible repeated games m of length k when these two players face each other. The average reward of m is $\mathcal{R}_n(m) = 1/k \sum_{i=1}^{k} u_n(z_i)$ and we denote $\mathbb{E}_{m \sim P^k_{\Omega_1,\Omega_2}}[\mathcal{R}_n(m)]$ to be the expected average reward when the players play k matches. From now on, if player n is not specified, we assume without loss of generality it is player 1. The proofs of the theorems have been omitted from this workshop version of the paper and can be found on ArXiV[2].

Definition 2. *Online algorithm Ω is a function $\mathcal{S}_n \times \Theta \mapsto \Delta(\mathcal{A}_n(s_n)) \times \Theta$, that maps an information state $s_n \in \mathcal{S}_n$ to*

[2]https://arxiv.org/abs/2006.08740

*the strategy $\sigma_n(s_n) \in \Delta(\mathcal{A}_n(s_n))$, while possibly making use of algorithm's state $\boldsymbol{\theta} \in \Theta$ and updating it. We denote the algorithm's **initial state** as $\boldsymbol{\theta}_0$. A special case of an online algorithm is a **stateless algorithm**, where the output of the function is independent of the algorithm's state (thus independent of the previous matches). If the output depends on the algorithm's state, we say the algorithm is **stateful**.*

As the game progresses, the online algorithm produces strategies for the visited information states and updates its algorithm state. This allows it to potentially output different strategies for the same information state visited in different matches. We thus use $\Omega^m(s_n)$ to denote the resulting strategy in the information state s_n after the algorithm has already played the matches $m = z_1, \ldots, z_k$. Note that players can not visit the same information state twice in a single match.

Remark 3. *If we need to encode a stochastic algorithm, we can do it formally as taking the initial state to be a realization of a random variable. The initial state should be extended to encode how the algorithm should act (seemingly) randomly in any possible game-play situation beforehand.*

Soundness of Online Algorithm

We are now ready to formalize the desirable properties of an online algorithm in our settings. Exploitability, resp. ϵ-equilibrium considers the expected utility of a fixed strategy against a worst-case adversary in a single match. We thus define a similar concept for the settings of an online algorithm in a repeated game: (k, ϵ)-soundness. Intuitively, an online algorithm is (k, ϵ)-sound if and only if it is guaranteed the same reward as if it followed a fixed ϵ-equilibrium after k matches.

Definition 4. *For an (k, ϵ)-sound online algorithm Ω, the expected average reward against any opponent is at least as good as if it followed an ϵ-Nash equilibrium fixed strategy σ for any number of matches k':*

$$\forall k' \geq k \, \forall \Omega_2 \, : \, \mathbb{E}_{m \sim P^{k'}_{\Omega,\Omega_2}} [\mathcal{R}(m)] \geq \mathbb{E}_{m \sim P^{k'}_{\sigma,\Omega_2}} [\mathcal{R}(m)] \, . \tag{1}$$

If algorithm Ω is (k, ϵ)-sound for $\forall k \geq 1$, we say the algorithm is ϵ-sound.

Note that this definition guarantees that an online algorithm that simply follows a fixed ϵ-equilibrium is ϵ-sound. And while the online algorithm can certainly play as a fixed strategy, online algorithms are far from limited to doing so, e.g. `PlayCache` from Section 3. `PlayCache` is 1-sound ($\epsilon = 1$) as this algorithm is highly exploitable in the first match. Additionally, an online algorithm may be sound ($\epsilon = 0$), but there might not be any offline equilibrium that produces the same distribution of matches.

Response Game

To compute the expected reward $\mathbb{E}_{m \sim P^{k'}_{\Omega,\Omega_2}} [\mathcal{R}(m)]$ as in Def. (4), we construct a repeated game (Osborne and Rubinstein 1994) in the FOG formalism, where we replace the decisions of the online algorithm with stochastic (chance) transitions. As we allow the online algorithm to be stateful and thus produce strategies depending on the game trajectory, the response game must also reflect this possibility. The resulting game $\mathcal{G}^k_\Omega$ is thus exponential in size as it reflects all possible trajectories of k matches. We call this single-player game a ***k*-step response game**.

The k-step response game allows us to compute the best response value of a worst-case adversary in k-match game-play. We will use overloaded notation $\mathit{brv}(\mathcal{G}^k_\Omega)$ to denote this value.

Theorem 5. *If $\forall k' \geq k \quad \mathit{brv}(\mathcal{G}^{k'}_\Omega) \leq k'\epsilon$, then algorithm Ω is (k, ϵ)-sound.*

Proof. If we used a fixed ϵ-equilibrium strategy σ in each match (repetition) of a response game $\mathcal{G}^{k'}_\sigma$, then the $\mathit{brv}(\mathcal{G}^{k'}_\sigma) = k'\epsilon$ because adversary can gain at most ϵ in each match. Since ϵ-sound algorithm should play at least as well as some offline ϵ-equilibrium, it must have $\mathit{brv}(\mathcal{G}^k_\Omega) \leq k\epsilon \; \forall k \geq 1$. For a (k, ϵ)-sound algorithm we add the condition of $\forall k' \geq k$. □

Tabularized Strategy

When an online algorithm produces the same strategy for an information state regardless of the previous matches, there is no need for the k-response game. Fixed strategy notion sufficiently describes the behavior of the online algorithm and thus the exploitability of the fixed strategy matches the soundness. To compute this fixed strategy, one simply queries the online algorithm for all the information states in the game.

4 Relating (k, ϵ)-Soundness and ϵ-Nash

Unfortunately, our notion of (k, ϵ)-soundness is often infeasible to reason about, as it requires checking that the algorithm does not make strategy errors for $\forall k' \geq k$. In this section, we introduce the concept of consistency that allows one to formally state that the online algorithm plays "consistently" with an ϵ-equilibrium. Our consistency notion allows us to directly bound the (k, ϵ)-soundness of an online algorithm. We introduce three hierarchical levels of consistency, with varying restrictions and corresponding bounds. Notice that they differ mainly in the order of quantifiers.

Local Consistency

Local consistency simply guarantees that every time we query the online algorithm, there is an ϵ-equilibrium that has the same local behavioral strategy $\sigma(s)$ for the queried state s.

Definition 6. *Algorithm Ω is locally consistent with ϵ-equilibria if*

$$\forall k \, \forall m = (z_1, z_2, \ldots, z_k) \;\; \forall h \in \mathcal{H}(z_k) \;\; \exists \sigma \in \mathcal{NE}^\epsilon_n$$
$$\textit{holds that} \;\; \Omega^{(z_1, \ldots, z_{k-1})}(s(h)) = \sigma(s(h)).$$

While this suggests that the algorithm plays like some equilibrium, it is not so, and the resulting strategy can be highly exploitable. This is because one cannot combine local behavioral strategies from different ϵ-equilibria and hope to

preserve their exploitability. In another perspective, as soon as one starts to condition the selection of the strategy on private information, it risks computing strategies that can be exploited in a repeated game. This is a motivation behind introducing (k, ϵ)-soundness, as it allows us to analyze algorithms that use such conditioning.

Consider the CMP game with two strategies $\sigma^1 = \{(s_1, p = 1), (s_2, q = 0)\}$ and $\sigma^2 = \{(s_1, p = 0.5), (s_2, q = 0.5)\}$. While both strategies are equilibria, if one plays in the states s_1 and s_2 based on the first and second equilibrium respectively, it corresponds to an exploitable strategy $\{(s_1, p = 1), (s_2, q = 0.5)\}$.

Theorem 7. *An algorithm that is locally consistent with ϵ-equilibria might not be (k, ϵ)-sound.*

Note that this can happen even in perfect information games. Interestingly, local consistency is sufficient if the algorithm is consistent with a subgame perfect equilibrium.

Theorem 8. *In perfect information games, an algorithm that is locally consistent with a subgame perfect equilibrium is sound.*

A particularly interesting example of an algorithm that is only locally consistent is Online Outcome Sampling (Lisý, Lanctot, and Bowling 2015) (OOS). See Section 6 for detailed discussion and experimental evaluation, where we show that this algorithm can produce highly exploitable strategies in imperfect information games.

Global Consistency

Local consistency guarantees consistency only for individual states. The problem we have then seen is that the combination of these local strategies might produce highly exploitable overall strategy. A natural extension is then to guarantee consistency with some equilibria for all the states in combination: a global consistency.

Definition 9. *Algorithm Ω is globally consistent with ϵ-equilibria if*

$$\forall k\ \forall m = (z_1, z_2, \ldots, z_k)\ \exists \sigma \in \mathcal{NE}^{\epsilon}_n\ \forall h \in \mathcal{H}(z_i)$$

holds that $\Omega^{(z_1, \ldots, z_{i-1})}(s(h)) = \sigma(s(h))$ *for* $\forall i \in \{1, \ldots, k\}$.

However:

Theorem 10. *An algorithm that is globally consistent with ϵ-equilibria might not be ϵ-sound.*

Proof. A counter-example: The `PlayCache` algorithm is globally consistent, but it is not sound ($\epsilon = 0$), as we have seen that it is exploitable during the first match ($k = 1$). □

But what if the algorithm keeps on playing the repeated game? While the global consistency with equilibria does not guarantee soundness, it guarantees that the expected average reward converges to the game value in the limit.

Theorem 11. *For an algorithm Ω that is globally consistent with ϵ-equilibria,*

$$\forall k\ \forall \Omega_2 : \mathbb{E}_{m \sim P^k_{\Omega, \Omega_2}}[\mathcal{R}(m)] \geq u^* - \epsilon - \frac{|\mathcal{S}_1|\Delta}{k}. \quad (2)$$

Corollary 12. *An algorithm Ω that is globally consistent with ϵ-equilibria is (k, ϵ)-sound as $k \to \infty$.*

Strong Global Consistency

The problem with global consistency is that it guarantees the existence of consistent equilibrium for any game-play *after* the game-play is generated. Strong global consistency additionally guarantees that the game-play *itself* is generated consistently with an equilibrium; and as in global consistency, the partial strategies for this game-play also correspond to an ϵ-equilibrium. In other words, the online algorithm simply exactly follows a predefined equilibrium.

Definition 13. *Online algorithm Ω is strongly globally consistent with ϵ-equilibrium if*

$$\exists \sigma \in \mathcal{NE}^{\epsilon}_n\ \forall k\ \forall m = (z_1, z_2, \ldots, z_k)\ \forall h \in \mathcal{H}(z_k)$$

holds that $\Omega^{(z_1, \ldots, z_{k-1})}(s(h)) = \sigma(s(h))$.

Strong global consistency guarantees that the algorithm can be tabularized, and the exploitability of the tabularized strategy matches ϵ-soundness of the online algorithm.

Theorem 14. *Online algorithm Ω that is strongly globally consistent with ϵ-equilibrium is ϵ-sound.*

Canonical examples of strongly globally consistent online algorithms are DeepStack/Libratus. In general, an algorithm that uses a notion of safe (continual) resolving is strongly globally consistent as it essentially re-solves some ϵ-equilibrium (albeit an unknown one) that it follows. Another, more recent example is ReBeL (Brown et al. 2020), as it essentially imitates CFR-D iterations in conjunction with a neural network.

Proving Strong Global Consistency

While we are not aware of an algorithm that is only globally consistent (besides the toy `PlayCache`), reasoning about global consistency can be beneficial for showing the strong global consistency. Doing so just based on its definition might not be straightforward. However, proving global consistency can be easier. If applicable, we can then use the following theorem to extend the proof to the strong global consistency.

Theorem 15. *If a globally consistent algorithm is stateless then it is also strongly globally consistent.*

Proof. The definition of a stateless algorithm implies that for an information state s the algorithm always produces the same behavioral strategy $\sigma(s)$ as the algorithm is deterministic (all stochasticity is encoded within the algorithm state $\boldsymbol{\theta}$, see Remark 3).

This means that whatever ϵ-equilibria the algorithm is globally consistent with is independent of the current game-play or match number. This allows us to swap the quantifiers from

$$\forall k\ \forall m = (z_1, z_2, \ldots, z_k)\ \exists \sigma \in \mathcal{NE}^{\epsilon}_n\ \forall i \in \{1, \ldots, k\}\ \forall h \in \mathcal{H}(z_i) :$$
$$\Omega^{(z_1, \ldots, z_{i-1})}(s(h)) = \sigma(s(h))$$

to

$$\exists \sigma \in \mathcal{NE}^{\epsilon}_n\ \forall k\ \forall m = (z_1, z_2, \ldots, z_k)\ \forall i \in \{1, \ldots, k\}\ \forall h \in \mathcal{H}(z_i) :$$
$$\Omega^{(z_1, \ldots, z_{i-1})}(s(h)) = \sigma(s(h)).$$

Using the same argument we can treat the different matches z_i as an iteration over k, leading us to strong global consistency

$$\exists \sigma \in \mathcal{NE}_n^\epsilon \; \forall k \; \forall m = (z_1, z_2, \ldots, z_k) \; \forall h \in \mathcal{H}(z_k) \; : \\ \Omega^{(z_1, \ldots, z_{k-1})}(s(h)) = \sigma(s(h)).$$

□

5 Relating (k, ϵ)-Soundness and Regret

Regret is an online learning concept that has triggered design of a family of powerful learning algorithms. Indeed, many algorithms that approximate Nash equilibria use regret minimization (Zinkevich et al. 2008). There is a well-known connection between regret and the Nash equilibrium solution concept. In a zero-sum game at time k, if both players' overall regret R_k is less than $k\epsilon$, the average strategy profile is a 2ϵ-equilibrium (Zinkevich et al. 2008). The use of k in (k, ϵ)-soundness allows us to relate it with regret, and show how it is different from the consistency hierarchy.

Corollary 16. *Any regret minimizer with a regret bound of R_k is $(k, \frac{R_k}{2k})$-sound.*

6 Experiments

A particularly interesting example of an algorithm that is only locally consistent is Online Outcome Sampling (OOS) (Lisý, Lanctot, and Bowling 2015). We use it to demonstrate the theoretical ideas in this paper with empirical experiments. We show that local consistency does in fact fail to result in ϵ-soundness in the online setting. The problem we demonstrate is also not specific to OOS, but in general to any adaptation of an offline algorithm to the online setting where the algorithm attempts to improve its strategy during online play.

At high level, OOS runs the offline MCCFR algorithm in the full game (while also gradually building the tree), parameterized to increase the sampling probability of the current information state. The algorithm then plays based on the resulting strategy for that particular state. The problem is that these individual MCCFR runs can converge to different ϵ-equilibria as the MCCFR is parameterized differently in each information state. In other words, the OOS algorithm exactly suffers from the fact that it is only locally consistent.

We use two games in our experiments: Coordinated Matching Pennies from Section 3 and Kuhn Poker (Kuhn 1950). We present the Coordinated Matching Pennies results here. The code for all the experiments can be found at https://github.com/deepmind/open_spiel/.

Within a single match of Coordinated Matching Pennies, the second player will act either in s_1 or s_2. OOS will therefore bias MCCFR samples to whichever information state that actually occurs in the match. These two situations are distinct and result in two different strategies for the whole game (including the non-visited state), similarly to the example in Section 4. To emulate what OOS does, we parametrize MCCFR runs to bias samples into s_1 and s_2 respectively, and initialize the regrets in s_1, s_2 so that the MCCFR is likely to produce diverse sets of strategies. As

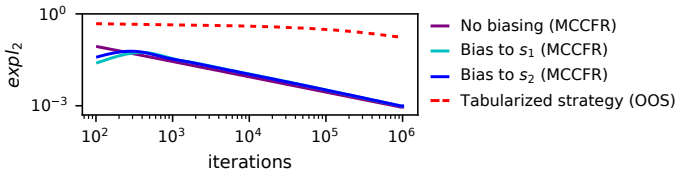

Figure 2: While individual MCCFR strategies have low exploitability of $\sim 10^{-3}$, the tabularized OOS strategy has high exploitability of 0.17 even after 10^6 iterations.

MCCFR is stochastic, we average the strategies over $3 \cdot 10^4$ random seeds.

In Figure 2 we plot exploitability for the average strategies, and unbiased MCCFR for reference. The two biased variants of MCCFR actually converge at a similar rate to unbiased MCCFR, confirming that OOS is locally consistent: it quickly converges to an ϵ-equilibria for s_1 and s_2 individually. However, the tabularized strategy — the strategy OOS follows online — is many orders of magnitude more exploitable even with hundreds of thousands of online iterations. The problem is that adapting its strategy online at s_1 and s_2 causes it to not be globally consistent with any ϵ-equilibria.

7 Related literature

There are several known pathologies that occur in imperfect information games that are not present in the perfect information case. The pathologies that happen in the offline setting also present a problem in the online setting. In (Frank and Basin 1998) the authors identified two problems: strategy-fusion and non-locality.

These two problems can easily arise for algorithms designed to solve only perfect-information games, such as minimax or reinforcement learning algorithms, and lead to computation of exploitable strategies. The proposed local consistency is similar in its spirit to non-locality, as composition of partial strategies (that correspond to parts of distinct equilibria) produced by an online algorithm may not be an overall equilibrium strategy. However local consistency identifies sub-optimal play also across repeated games.

In (Moravcik et al. 2017; Brown and Sandholm 2018), they use some form of continual re-solving, which is strongly globally consistent. This guarantees soundness of the algorithms. Continual resolving uses value functions defined over public belief spaces (Brown et al. 2020) to compute consistent strategies. Indeed, the minimal amount of information needed to properly define value functions are ranges (beliefs) over common knowledge public states (Seitz et al. 2019).

The notion of sufficient plan-time statistics studied in (**?**) is very closely related to the public beliefs. The paper suggests the structure of the value function for games where the hidden information becomes public after a certain number of moves.

We are not aware of algorithms in the literature that are only globally consistent. This may lead to interesting future work: the algorithm may try to reduce its sub-optimal play of

the first matches, while possibly not using all of the required player's ranges.

Tabularization has been used in (Šustr, Kovařík, and Lisý 2019) to compute an offline strategy and its exploitability. In (Lisý, Lanctot, and Bowling 2015) they consider computing this tabularization (they refer to it as "brute-force" approach), but it is a very expensive procedure. Instead they use an "aggregate method", which "stitches" strategy from a small number of matches and defines the strategy as uniform in non-visited information states. They do not state whether such approximation of tabularization is indeed correct.

8 Conclusion

We introduced the game of Coordinated Matching Pennies (CMP). This game illustrates the consistency issues that can arise for online algorithms in imperfect information games. We observed that exploitability is not an appropriate measure of an algorithm's performance in online settings. This motivated us to introduce a formal framework for studying online algorithms and allowed us to define ϵ-soundness. Just like ϵ-exploitability, it measures the performance against the worst-case adversary. Soundness generalizes exploitability to repeated sequential games and it collapses to it when an online algorithm follows a fixed strategy. We then introduced a hierarchical consistency framework that formalizes in what sense an online algorithm can be consistent with a fixed strategy. Namely, we introduced three levels of consistency: i) local, ii) global and iii) strongly global. These connect an online algorithm's behavior to that of a fixed strategy with increasingly tight bounds on the average expected utility against a worst-case adversary. We also stated various bounds on soundness based on the exploitability of a consistent fixed strategy. Interestingly, the implications are different in some cases for perfect and imperfect information games.

Within this framework, we saw that local consistency in imperfect information games does not guarantee correct evaluation of worst-case performance by computing exploitability. Based on this result, we argued that OOS, previously considered sound, can be exploited. This illustrates that these subtle problems with online algorithms can easily be missed and lead to wrong conclusions about their performance. Our experimental section included experiments in CMP and Kuhn Poker and showed a large discrepancy between OOS's actual performance and the bound previously thought to hold.

References

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. *arXiv preprint arXiv:2007.13544* .

Brown, N.; and Sandholm, T. 2017. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems*, 689–699.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359(6374): 418–424.

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science* 365(6456): 885–890.

Burch, N.; Johanson, M.; and Bowling, M. 2014. Solving imperfect information games using decomposition. In *AAAI*, 602–608.

Campbell, M.; Hoane Jr, A. J.; and Hsu, F.-h. 2002. Deep blue. *Artificial intelligence* 134(1-2): 57–83.

Copeland, B. J. 2004. *The essential Turing*. Clarendon Press.

Frank, I.; and Basin, D. 1998. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence* 100(1-2): 87–123.

Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisý, V. 2019. Rethinking formal models of partially observable multiagent decision making. *arXiv preprint arXiv:1906.11110* .

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

Lisý, V.; Lanctot, M.; and Bowling, M. 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*, 27–36. International Foundation for Autonomous Agents and Multiagent Systems.

Moravcik, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337): 508–513.

Osborne, M. J.; and Rubinstein, A. 1994. *A course in game theory*. MIT press.

Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development* 3(3): 210–229.

Seitz, D.; Kovarík, V.; Lisỳ, V.; Rudolf, J.; Sun, S.; and Ha, K. 2019. Value Functions for Depth-Limited Solving in Imperfect-Information Games beyond Poker. *arXiv preprint arXiv:1906.06412* .

Shannon, C. E. 1950. XXII. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41(314): 256–275.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354–359.

Šustr, M.; Kovařík, V.; and Lisý, V. 2019. Monte Carlo continual resolving for online strategy computation in imperfect information games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 224–232. International Foundation for Autonomous Agents and Multiagent Systems.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.