Learning in Games with Progressive Hiding

Benjamin Heymann
Criteo AI Lab
Paris, France
b.heymann@criteo.com

Marc Lanctot
Google DeepMind
Montreal, Canada
lanctot@google.com

ABSTRACT

When learning to play an imperfect information game, it is often easier to first start with the basic mechanics of the game rules. For example, one can play several example rounds with private cards revealed to all players to better understand the basic actions and their effects. Building on this intuition, this paper introduces *progressive hiding*, an algorithm that balances learning the basic mechanics of an imperfect information game and satisfying the information constraints. Progressive hiding is inspired by methods from stochastic multistage optimization, such as scenario decomposition and progressive hedging. We prove that it enables the adaptation of counterfactual regret minimization to games where perfect recall is not satisfied. Numerical experiments illustrate that progressive hiding produces notable improvements in several settings.

KEYWORDS

Computational Game Theory, Information Relaxation, Counterfactual Regret Minimization

ACM Reference Format:
Benjamin Heymann and Marc Lanctot. 2025. Learning in Games with Progressive Hiding. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

This paper shows how the learning process in games with imperfect information can be improved by relaxing the information constraints with penalty-like methods. For a game theorist, *games* are mathematical models to describe strategic interactions between agents (the players of the game). Games, in their standard acceptance, which usually supports this definition, are often referred to as the "Drosophila of Artificial Intelligence" [25].[1] This comparison with the fruit fly comes from the fact that games provide Artificial Intelligence (AI) researchers with standardized, easily reproducible environments where they can test and benchmark various AI techniques. Games offer challenges that are complex enough to be interesting for AI research, but still have well-defined rules. One of the longstanding AI research questions about games is how to learn good strategies.

The field has advanced significantly due to breakthroughs where AI algorithms have outperformed human players in strategic games, illustrating the potent applications of AI in complex decision-making environments. Notable examples include Chess, Go, Poker and Stratego [3, 6, 7, 26, 30, 36, 39].

While Chess and Go are perfect information games, Stratego and Poker are imperfect information games, in the sense that the state of the game is not fully observed by the players. Hence players must account for the private information held by other players, as well as how their actions reveal their private information. Notably, this makes room for elements like bluffing [42]. Regret minimization at scale played a key role in recent achievements in AI for imperfect information games [6, 7, 26].

The AI community recently identified the card game Hanabi as a challenge for AI [1]. What differentiates Hanabi from the previously mentioned games is that it is a cooperative game[2]: all players have the same objective. The difficulty comes from the uncertainty of the draws and from the fact that the players cannot communicate freely. The problem of pursuing a common objective with limited communication is not new, and spans several fields, in particular economic theory, control, game theory, and machine learning [9, 15, 16, 19, 22, 27, 40, 44, 45]. Witsenhausen showed, with his landmark counterexample [44], how imposing communication constraints on a problem drastically changes the nature of the problem.

We present a general framework for dealing with information in games. The framework is best described with the following real life observation: when teaching a child how to play a game — like Poker or Hanabi — with hidden information, one often starts with a few plays where all the cards are revealed, and then, as the child is learning, one starts hiding more and more information to make the game more challenging. This metaphor summarizes the idea of the main algorithm presented in this paper. Because of this metaphor— and also as a tribute to Rockafellar and Wets' *progressive hedging* algorithm [35][3]— we name the algorithm *progressive hiding*.

A common assumption in games is that players remember what they see and what they do in the precise order of occurrence. In his seminal work [18] on the tree representation of extensive form games, this assumption is what Kuhn referred to as **perfect recall**. One difficulty with a game such as Hanabi is that, if one wants to see the whole group of players as one player, a *team*, then this *team* does not satisfy the perfect recall assumption because this one (big) player that constitutes the team is not allowed to remember the private information from its past. This absence of perfect recall prevents us from using key learning techniques [19, 21] such as counterfactual regret minimization (CFR) [46]. As we show, progressive hiding's modification of the game allows recovering some notion of perfect recall, which makes the new problem amenable to CFR. In an effort to bridge ideas from decentralized control theory and computational game theory, we depart from the customary tree notations and introduce a hybrid between product games [15] and games on trees.

[1] The fruit fly is widely used in experiments due to its ease of cultivation in large numbers outside its natural habitat and its rapid reproduction cycle.

[2] the term same payoff games is also very often used

[3] See Section 3 for related work.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

This new, compact notation reveals quite practical for our purpose. While product games were fully formalized relatively recently, they are just the continuation of an idea that was hinted half a century ago by Witsenhausen [45], specifically as a way to deal with setting such as ours (communication constraints).

1.1 Contribution

Our main contribution – Theorem 2 – is to show how information relaxation can be made compatible with learning in games. Information relaxation has never been used in imperfect information games, despite the fact that it is very close to how people discover how to play games in real life.

While we mostly focus on same payoff settings in this paper, we believe progressive hiding and information relaxation methods to be full of potential for the competitive case too (as opposed to same payoff games). The reason is similar to the one provided in the paper on progressive hedging [35] for solving stochastic dynamic programs: take a game for which a large part of the complexity comes from the hidden information (think Stratego or Poker or Bridge). Without the hidden information, those games could be amenable to deterministic methods (dynamic programming, for instance). Progressive hiding provides a way to navigate between the two worlds (deterministic and stochastic). In this work, we deliberately focus on same payoff games because we believe Theorem 2 illustrates well the strength of the general philosophy of information relaxation.

Section 2 introduces some background, including the product game notation, that will prove practical for our purpose. Section 3 is an attempt to make a historical connection between algorithmic game theory and a class of stochastic programming technique called scenario decomposition (e.g. progressive hedging). Section 4 presents a penalty method for games that can be analyzed under the lens of the proximal algorithms. This method is a first step toward our proposal, and brings the optimization tools that will be assembled with the learning ones in the second step. We then introduce in Section 5 progressive hiding (Algorithm 2), an algorithm that combines no-regret learning and information-relaxation. We show, in particular, how this approach allows us to recover the main property of CFR in an auxiliary game. We document numerical experiments in Section 6. Section 7 provides discussion on the limits of Algorithm 2. All proofs and code are provided in the arXiv version of the paper.

2 PRELIMINARY

This section is dedicated to background material on learning in games. There are different ways of modeling extensive form games [32], typically modeled as game trees. In this paper, we borrow ideas from the product form [15] as it better fits our needs for modeling the information structure. Games in product form represent information over a product set.

An important property of product games is that modifying the information structure of a product game can be done without modifying the product set over which this product game is defined. Such property is not satisfied by the tree representation (see Appendix on *lumberjacking*). Also, the product form provides a common framework for game theory and causality theory [14].

In what follows, if S is a set and $s \in S$, we will use the notation $-s$ to refer to the set $\{s' \in S, s' \neq s\}$ when it is clear what S is from the context. If n is an integer, then we set $[n] = \{1, 2, \dots, n\}$.

2.1 Motivation for using the product form

In an extensive form game (EFG), the order of play is encoded in a tree structure. This notation originates from [18] and is used extensively in game theory. In this paper, we use the more general formalism of product games. Product games shift the focus from the temporal order of play to the information available at each decision nodes. The game is supported by a Cartesian product instead of being supported by a tree. The reason to use this generalization of EFG is the need for flexibility when we modify the information structure of the game when doing information relaxation.

For example, suppose we discover at the end of the game if a coin toss was head or tail. Then we might end-up with a different tree representation when we allow the player to observe the coin toss result at the beginning of the game. Similarly, if one consider a game with hidden actions, and several possible variations where some of those hidden actions are revealed. Then the resulting trees will differ in their structures. For instance, if Bob and Alice play Rock-Paper-Scissors, a tree that represent a modified game where Bob observes Alice action (Alice plays first, Bob second) will not be the same as a tree where Alice sees Bob action (Bob plays first, Alice second).

In this work, the product formalism allows to abstract away from the order of play encoded in the tree representation. It also allows defining useful objects such as information maps and projectors in a very compact manner. Compared to general extensive-form games as in Osborne and Rubinstein [28, Chapter 11], Nature plays *only once* at the very beginning, and the random state is revealed progressively to the players. The order of play does not need to be known in advance, as it is the case for games on tree. This choice of formalism is made without loss of generality and based on convenience for the problem considered in this paper.

2.2 Extensive Game in Product Form

We first introduce a minimalistic version of the product form [4] for extensive form games, which is an alternative to the tree representation. The appendix contains more comprehensive information on the product game formalism. In what follows, P is the set of players in the game, L is a positive integer that represents the maximum number of turns in the game (*i.e.*, maximum length of any game), and W is a positive integer that represents the maximum number of legal actions at any time. We denote by Δ the W-dimensional simplex. We encode the available information to a player with an abstract, finite set $\mathbb{G}$. Otherwise said, $\mathbb{G}$ is a finite set that is used to encode the players' information as the game proceeds. The randomness owned by the game itself (as opposed to the randomness controlled by the players), sometimes referred to as **Nature's moves**, is encoded with a discrete probability space $(\Omega, \mathbb{P})$. The set Ω represents the space where random events take place. For example, in a card game, Ω would be the set of all possible orders of cards in the deck. In a dice game, it would be the set of all possible sequences of dice rolls during the game. There is a probability measure on Ω, denoted by

[4] without explicit references to the *information algebras* and the *solution map*

$\mathbb{P}$. For instance, in Poker, all possible orders of cards are equally probable. Last we define the **product set** as $\mathbb{H} = \Omega \times [W]^L$.

Given the primitives already introduced, a **game in product form** consists of a tuple $(\mathcal{P}, \mathcal{A}, \mathcal{X}, r)$ where: $\mathcal{P} : [L] \to P$ is a map that indicates which player needs to play at stage $i \in [L]$; $\mathcal{A} : \mathbb{G} \to [W]$ is a map that encodes the number of available actions at each stage of the game; for all $i \in [L]$, a map $\mathcal{X}_i : \mathbb{H} \to \mathbb{G}$ that encodes the available information at each stage of the game, and that only depends on the first i components of $\mathbb{H}$ (Nature's move and the predecessors);[5] the map $r : \mathbb{H} \to \mathbb{R}^P$ encodes the reward of the game, more precisely, r_p is the reward of player p. For any element in $\mathbb{H}$, we denote by $h_\emptyset$ its first component (associated with Nature's moves) and by h_i the $(i+1)^{th}$ component, associated with the i^{th} move of the players.

2.3 Example of game in product form

The product form for Cooperative Matching Pennies (Figure 2) involves one (team)-player represented by Alice and Bob. Nature has two equiprobable moves in $\Omega = (=, \neq)$. Alice and Bob pick their decisions in (H, T) and (H, T, P), respectively. So the product of Nature's set and the action sets is $\mathbb{H} = (=, \neq) \times (H, T) \times (H, T, P)$. Bob only observes Alice's move. Formally, for

$$h_1 = \left(\omega^1, d^1_{\text{Alice}}, d^1_{\text{Bob}}\right) \quad \text{and} \quad h_2 = \left(\omega^2, d^2_{\text{Alice}}, d^2_{\text{Bob}}\right),$$

we have $\mathcal{X}_{\text{Bob}}(h_1) = \mathcal{X}_{\text{Bob}}(h_2)$ if $d^1_{\text{Alice}} = d^2_{\text{Alice}}$, and since Alice only sees Nature's decision: $\mathcal{X}_{\text{Alice}}(h_1) = \mathcal{X}_{\text{Alice}}(h_2) \iff \omega^1 = \omega^2$. We do not care about the codomain of $\mathcal{X}$ as long as we can encode this type of relation. The seminal paper on product games [15] does not use the information map $\mathcal{X}$, but this requires manipulating more abstract objects (e.g., σ-fields). Alice's policy on $\mathbb{H}$ can be reduced to a function on Ω via the information map. Similarly, Bob's policy can be represented by a function whose domain is Alice's decision set. Given such policies, sampling from $(=, \neq)$, Alice's policy, and Bob's policy produces a sequence in $\mathbb{H}$ with distribution Q_μ that we present in Section 2.4.

2.4 Policies and Push-Forward Probability

Next, building on the primitives we introduced in Section 2.2, we can now describe how a product game is played by specifying the policies and the resulting push-forward probability. For a decision time $i \in [L]$, we denote by $\bar{\Lambda}_i$ the set of (admissible) policies μ_i, where $\mu_i : \Omega \times [W]^{i-1} \to \Delta$ and $\operatorname{supp} \mu_i(h) \subset [\mathcal{A} \circ \mathcal{X}_i(h)]$, that is, the policy only selects admissible actions. Using a canonical extension, we can identify μ_i with a map from $\mathbb{H} \to \Delta$ that depends at most on Nature's move and the first $i-1$ decisions. For $i \in [L]$, an **implementable** policy $\mu_i \in \bar{\Lambda}_i$ should also satisfy, for any histories h and h' in the product space $\mathbb{H}$

$$\mathcal{X}_i(h) = \mathcal{X}_i(h') \implies \mu_i(h) = \mu_i(h'). \tag{1}$$

In words, an implementable policy depends solely on what player $\mathcal{P}(i)$ is supposed to know at the time of the decision. Relation (1) is the **non-anticipativity constraint** [37], and indicates that the policy μ_i should only depend on the information given by $\mathcal{X}_i$ to player $\mathcal{P}(i)$ at step i. If a policy is not implementable, it means that it needs some pieces of information that are not available at the time of the decision. We denote by Λ_i the subset of policies from $\bar{\Lambda}_i$ that are implementable. Also, we denote by Λ_p and $\bar{\Lambda}_p$ the sets of implementable and admissible policy profiles of player p for $p \in [P]$: $\Lambda_p = \times_{i, \mathcal{P}(i)=p} \Lambda_i$. We similarly define Λ and $\bar{\Lambda}$ to refer to policy profiles of all the players: $\Lambda = \times_p \Lambda_p$.

The detail of the construction of a game in product form goes back to Witsenhausen's seminal paper, and was then clarified in Heymann et al. [15], Witsenhausen [45]. We recognize that the construction might feel uneasy for readers used to the tree formulation. We refer to [15, 45] for more details on this construction.

How do we play a game in product form? Given a deterministic policy profile μ and an element ω of Ω, there is a unique element h of $\mathbb{H}$ that satisfies $(\omega, \mu(h)) = h$. We pinpoint that this relation is on the full strategy profile (on all the players, not only on a player of interest). The relation states that if we look at a realization of a game (all the decisions), and replay the decisions along the path using the strategies that were used for this realization, we should recover the realization itself. It is a well-posedness condition. Typically, one wants to exclude temporal paradoxes. In a game on tree, one usually first describes the game with deterministic policies and then specifies a way for the players to randomize. Here it is the same, the well-posedness of the game is checked on a deterministic specification. Then [15] provide constructions to extend the space of pure strategies to randomized strategies.

Given a policy profile $(\mu_i)_{i \in [L]}$, we can therefore construct the associated push-forward probability[6], denoted by $\mathbb{Q}_\mu$, on the product space $\mathbb{H}$. It is a probability on the realizations of the game. It obviously depends on the strategies of the players μ. We then denote by $\mathbb{E}_\mu$ the **associated expectation operator**. Last, we define a notation for modifying policies that allows for easy adjustment of specific components. Let μ be a base policy, and consider the operation of altering its i-th component to a new value k. This modification is denoted by $\mu(i \to k)$. [7]

An ϵ-Nash equilibrium is an admissible strategy profile μ so that no player can improve her outcome by more than ϵ from an unilateral deviation, that is, for any player p and any other admissible strategy profile μ'_p, $\mathbb{E}_{\mu_p, \mu_{-p}}[r_p(\mathbf{h})] + \epsilon \geq \mathbb{E}_{\mu'_p, \mu_{-p}}[r_p(\mathbf{h})]$. A Nash equilibrium is a 0-Nash equilibrium.

2.5 Perfect Recall and Information Maps

Perfect recall is a standard assumption in extensive-form games [38, 46]. An important property of perfect recall is that it implies the equivalence between mixed and behavioral strategies [15, 18]. In words, perfect recall means that the player has perfect memory of what they see and do and the precise order that information was revealed over the turns. In our context, perfect recall for player p corresponds

[5] this assumption can be weakened, but helps the presentation

[6] The term push-forward comes from probability theory.

[7] Formally, the modified policy $\mu(i \to k)$ is given by

$$\mu(i \to k)_j = \begin{cases} k & \text{if } j = i, \\ \mu_j & \text{otherwise.} \end{cases}$$

This notation can be extended to accommodate multiple simultaneous modifications. For instance, changing the i-th component to k and the j-th component to l in μ is expressed as $\mu(i \to k)(j \to l) = \mu(i \to k, j \to l)$.

to the two conditions, for any $i < j$ such that $\mathcal{P}(i) = \mathcal{P}(j) = p$

$$\left(\mathcal{X}_i(h) \neq \mathcal{X}_i(h') \implies \mathcal{X}_j(h) \neq \mathcal{X}_j(h')\right) \tag{2}$$

$$\text{and } \left(h_i \neq h'_i \implies \mathcal{X}_j(h) \neq \mathcal{X}_j(h')\right). \tag{3}$$

The first condition states that if the information known about h and h' differs on turn i, then this must remain true at a future turn $j > i$. The second condition states that if the action taken at turn i is different, then the player must remember this information at a later turn $j > i$. While the perfect recall assumption is realistic when the player corresponds to a single agent, this is less so when the player is used to model a team of several agents. The absence of perfect recall in a game prevents the use of several tools, for instance, backward induction, CFR [19, 46], and the equivalence of behavioral and mixed policies [18].

We say that an information map $\mathcal{X}_i^+$ is *finer* than an information map $\mathcal{X}_i$ if for any $h \in \mathbb{H}$ $\mathcal{X}_i(h) \neq \mathcal{X}_i(h') \implies \mathcal{X}_i^+(h) \neq \mathcal{X}_i^+(h')$. If $\mathcal{X}_i^+$ is finer than $\mathcal{X}_i$, we can also say equivalently that $\mathcal{X}_i$ is *coarser* than $\mathcal{X}_i^+$. We say that $\Lambda^+ \subset \bar{\Lambda}$ *is induced* by an information map profile $\mathcal{X}^+$ if it is the set of policy profiles from $\bar{\Lambda}$ that are implementable with respect to $\mathcal{X}^+$. When we take a finer information map, the game become in some sense easier, because there is less uncertainty. The idea of information relaxation is to take a finer information map to help the learning process, and then penalize the usage of the free information provided. An important tool to do so is the projector, which we define next.

2.6 Projector

Let μ be any full support implementable policy profile, and $\tilde{\mathbb{H}}$ be the support of $\mathbb{Q}_\mu$. For $\mu_0 \in \bar{\Lambda}$ of full support, we define $\text{Proj}_{\mu_0}(\mu)$ as the element γ of $\bar{\Lambda}$ such that

$$\gamma_i(h) = \mathbb{E}_{\mu_0}[\mu_i(\mathbf{h'})|\mathcal{X}_i(\mathbf{h'}) = \mathcal{X}_i(h)] \quad \forall i \in [L] \quad \forall h \in \tilde{\mathbb{H}}. \tag{4}$$

Suppose μ_i was defined using an information map finer than $\mathcal{X}$, then, by property of the condition expectation, the projector transforms μ_i into an implementable policy for $\mathcal{X}_i$.

We have the following key properties (adapted from [35]):

Proposition 1. $\forall \mu \in \bar{\Lambda}, \mu \in \Lambda \iff \text{Proj}_{\mu_0}(\mu) = \mu$.

Proposition 2. $\text{Proj}_{\mu_0} \circ \text{Proj}_{\mu_0} = \text{Proj}_{\mu_0}$.

Notably, proposition 1 provides an **alternative representation of the non-anticipativity** constraint (1).

2.7 No-regret learning in games

No-regret learning. The theory of online learning envision a decision maker that makes sequential decisions x_t in a convex, compact set $\mathbb{X}$ at each epoch $t = 1 \dots T$, afterwhat the (possibly adversarial) environment outputs a concave reward ℓ_t. An important metric of success for an algorithm is the regret, defined as $\max_{x \in \mathbb{X}} \sum_{t=1}^T \ell_t(x) - \sum_{t=1}^T \ell_t(x_t)$. A regret minimization algorithm is one where this regret grows at a rate of $o(T)$. An algorithm with this property is called *Hannan-consistent*. Regret minimization algorithms play a central role in optimization and game theory, because they can be used to maximize a function or find approximate Nash equilibrium [2]. For example, in the zero-sum two-player setting, one can find ϵ-Nash equilibrium by repeating the game with two Hannan-consistent learners.

CFR. We next describe counterfactual regret minimization (CFR), which was introduced in [46] and extended in many papers, in particular [17, 19, 21, 23]. For $k \in [W]$ denote by δ_k the constant policy that always select action k. The CFR algorithm keeps a collection of local regret minimizers indexed by $\{(i, g) \in [L] \times \mathbb{G}; g \in \mathcal{X}_i(\tilde{\mathbb{H}})\}$. Then at each step, the regret minimizer (i, g) is fed with the reward vector of size $[\mathcal{A}(g)]$ and of kth component $\mathbb{E}_{\delta_k, \mu_{-i}^t}[r_{\mathcal{P}(i)}(\mathbf{h})|\mathcal{X}_i(\mathbf{h}) = g]$ for $k \in [\mathcal{A}(g)]$. The strength of CFR is that when a player satisfies the perfect recall assumption 2, their overall regret is upper-bounded by the sum of the regrets of the local regret minimizers.

3 RELATED WORKS

In this section, we attempt to draw a connection between algorithmic game theory and a stochastic programming technique called scenario decomposition. As argued in [35] a common methodology for addressing uncertainty involves initially adopting a deterministic simplification. Practitioners first solve the problem for some known scenarios, thereby establishing a baseline of deterministic solutions. Subsequently, these solutions are aggregated to incorporate uncertainty. This approach simplifies complex problems by breaking them down into more manageable components.

In game theory, this type of approach is sometimes branded as Perfect Information Monte Carlo search (PIMC search [24]). Motivated by the unexpected success of the methods in Bridge [12], Skat [8] and Hearts [43], the authors of [24], discuss conditions for PIMC to work. While the theoretical limit of the PIMC approach has been recognized very early [10], it still achieves state of the art performance in games such Skat and Bridge. Further improvements of PIMC are proposed in [11, 41].

Paraphrasing the authors of [35], the aim of progressive hedging is to provide "a rigorous algorithmic procedure for determining (...) a policy in response to any weighting of the scenarios". The algorithm iteratively tweaks the rewards in each scenarios so that at some point, the procedure finds a policy that is adapted to the uncertainty filtration. The algorithm leverages the ideas of augmented Lagrangian and proximal algorithms, and enjoys theoretical guaranties under some convexity assumptions. Progressive hedging was recently adapted to variational inequalities [34]. In the same vein, the recent monograph [37] presents Lagrangian methods applied to non-anticipativity constraints in the context of stochastic programming. The discretization of the random space and the introduction of stochastic dual variables allows decomposing stochastic programs per scenarios. The dual interpretation comes with important results from convex analysis [33].

Crucially, with these stochastic programming techniques, the decomposition is done per scenario, not per agent. To apply online learning techniques, we would prefer to obtain a decomposition per agent. Furthermore, in stochastic optimization, a sole decision maker optimizes jointly all the decisions. By contrast, in games with imperfect information, players do not observe the same information, and the action of a player can influence what the other players observe.

4 INFORMATION RELAXATION

In this section, we combine the penalty method and proximal algorithms [29] to produce a player best response, that is, a policy that is optimal given the policy of the other players. While the resulting Algorithm 1 might be of independent interest, it serves as a guiding principle for the design of progressive hiding, presented in the next section. We first reduce to the case where $|P| = 1$ by fixing the policies of all players but one. Let $\tilde{\Lambda} \subset \bar{\Lambda}$ induced by an information map profile $\tilde{\mathcal{X}}$ such that $\Lambda \subset \tilde{\Lambda}$. Instead of restricting our search to implementable policies, we could optimize over $\tilde{\Lambda}$. However, the optimal policy would very likely not satisfy the non-anticipativity constraint. Using Proposition 1, we suggest to favorize implementability through a penalty term $-\lambda||\mu - \text{Proj}(\mu)||^2$, where Proj is a projector as introduced in Section 2.6 induced by a policy μ_0 and $||\cdot||$ is the reweighted L_2 norm associated with μ_0. This idea drives us to a relaxed version of the problem of finding an implementable best response. Indeed, setting $\mathcal{L}(\mu, \lambda) = \mathbb{E}_\mu[r_p(\mathbf{h})] - \lambda||\mu - \text{Proj}(\mu)||^2$, we propose to solve the relaxed problem

$$\max_{\mu \in \tilde{\Lambda}} \mathcal{L}(\mu, \lambda). \tag{5}$$

Observe that, because of the penalty, the criterion is unusual and might be impractical for many game solving approaches. A first step to address this difficulty is Theorem 1, which states that one can produce a local optimum by composing two proximal steps, which can be analyzed under the lens of the Majorize-Minimization Algorithm (Algorithm 1). Progressive hiding builds on this idea by adding a linearization step as in [47] for online convex problems.

Algorithm 1: Resolution by Information Relaxation

Input: $\lambda > 0$, $\mu^{(0)} \in \bar{\Lambda}$, $T \in \mathbb{N}$
Initialization: $\mu = \mu^{(0)}$
for $t \leftarrow 1$ **to** T **do**
  $\gamma \leftarrow \text{Proj}(\mu)$ `// projection step`
  $\mu \leftarrow \arg\max_{\mu \in \tilde{\Lambda}} \mathbb{E}_\mu[r_p(\mathbf{h})] - \lambda||\mu - \gamma||^2$ `// proximal step`
endfor
return (μ, γ)

Theorem 1. *Fix $\lambda > 0$, $\mu^0 \in \bar{\Lambda}$ and denote by (μ^T, γ^T) the output of Algorithm 1, then $\mathcal{L}(\mu^T, \lambda)$ is non-decreasing and admits a limit.*

Proposition 3. *We have the relation*

$$\max_{\mu \in \Lambda} \mathbb{E}_\mu[r_p(\boldsymbol{h})] \leq \max_{\mu \in \tilde{\Lambda}} \mathbb{E}_\mu[r_p(\boldsymbol{h})] - \lambda||\mu - \text{Proj}(\mu)||^2. \tag{6}$$

The type of upper bound of Proposition 3 is often used to estimate duality gap in operations research [4]. Observe that when λ is small, Problem (5) becomes closer to the problem of finding an optimal policy in $\tilde{\Lambda}$, so the couple (μ, γ) produced by Algorithm 1 might not satisfy μ being close to γ, which means that μ might be far from implementable. By contrast, if λ is large, one expects μ and γ to be close, however, because of the proximal term, in each iteration only solutions very close to the previous steps will be considered, and we might get stuck with a local optima. In practice, many strategies could be used to solve the arg max step in Algorithm 1. In particular, if the auxiliary game (with $\tilde{\Lambda}$ instead of Λ) satisfies perfect recall, then one can rely on backward induction. The next section, which presents our main contribution, is guided by the idea of combining the philosophy of Algorithm 1 with regret minimization approaches.

5 PROGRESSIVE HIDING

We next present an algorithm — progressive hiding — that replicates the philosophy of information relaxation in the framework of no-regret learning by introducing an auxiliary game. This section also contains the article principal contribution, which is Theorem 2. We suppose the players in $-p = P \setminus \{p\}$ play according to a policy profile μ^t_{-p} at stage t.

5.1 Algorithm

We suppose we have access to a class of **low-regret algorithms** that implements two functions: Observe (to get the realization of vector of linear reward) and Decide (to output a probability distribution on the decision). Progressive hiding requires at each time t **a non-negative penalty parameter** λ^t for the non-anticipativity constraint violation. This sequence of parameters λ^t might be time adaptive to allow for more flexibility at the beginning, but still shift toward implementability at the end of the learning process. Progressive hiding also requires an **information map** $\tilde{\mathcal{X}}$, which is a refinement (cf. Section 2.5) over the information map $\mathcal{X}$. For example, $\tilde{\mathcal{X}}$ can inform the player about the hands of its opponents, or about what other teammates saw in the previous turns. The finer the information map $\tilde{\mathcal{X}}$, the more additional information is provided in the auxiliary game. Notably, ensuring that perfect recall is satisfied in the auxiliary game seems to be a fruitful direction, as indicated by Theorem 2. Last, progressive hiding also requires a sequence of **projectors** Proj^t as defined in Section 2.6. We set $\text{Proj}^t = \text{Proj}_{\mu^t_p, \mu^t_{-p}}$. We denote by I_p the elements i of $[L]$ such that $\mathcal{P}(i) = p$, and we set $G_i = \tilde{\mathcal{X}}_i(\mathbb{H})$, for $i \in I_p$. We introduce the notation $\mathbb{E}_\mu[\mathbf{r}_p|g] = \mathbb{E}_\mu[r_p(\mathbf{h})|g \in \{\tilde{\mathcal{X}}_i(\mathbf{h}), i \in I_p\}]$. Last, we introduce the time-dependent, **random penalty**, for $i \in I_p$ and $h \in \tilde{\mathbb{H}}$

$$\ell^t_i(\mu_i(h), \tilde{\mathcal{X}}_i(h)) = \lambda^t||\mu_i(\tilde{\mathcal{X}}_i(h)) - \text{Proj}^t(\mu^t_p)_i(\tilde{\mathcal{X}}_i(h))||^2. \tag{7}$$

The right-hand side of Equation (7) makes sense because (1) μ_i is a function of $\tilde{\mathcal{X}}_i(h)$, (2) $\text{Proj}^t(\mu^t_p)$ is a function of $\mathcal{X}_i(h)$ and $\tilde{\mathcal{X}}$ is finer than $\mathcal{X}$. Equation 7 corresponds to a penalization of the *lack of implementability* of the current solution.

Let μ^t be the iterates of progressive hiding defined in Algorithm 2. We denote respectively by

$$\varrho_t(\mu_p) = \mathbb{E}_{\mu_p, \mu^t_{-p}}\left[r_p(\mathbf{h}) - \sum_{i \in I_p} \ell^t_i(\mu_i(\mathbf{h}), \tilde{\mathcal{X}}_i(\mathbf{h}))\right],$$

$$R^T = 1/T \max_{\mu \in \tilde{\Lambda}} \sum_{t=1}^{T} \varrho_t(\mu_p) - \varrho_t(\mu^t_p),$$

the new criteria for the player of interest at a given time step t, and the average regret R^T in the auxiliary game with relaxed information constraints and penalty for violation. Also, for $i \in I_p$ and $g \in G_i$, we

set

$$\varrho_t[i,g][\mu_i] = \mathbb{E}_{\mu_p^t(i\to\mu_i),\mu_{-p}}\Bigg[r_p(\mathbf{h}) - \lambda^t\Big(2\langle \mu_i^t(\mathbf{h}) - \textsc{Proj}^t(\mu_p^t)_i(\mathbf{h}) \mid \mu_i\rangle\Big) - \sum_{i'>_p i} \ell_{i'}^t(\mu_{i'}^t(\mathbf{h}), \tilde{\mathcal{X}}_{i'}(\mathbf{h}))\Big| g\Bigg],$$

which corresponds to the *local* criterion optimized by progressive hiding, and $R_{loc}^T(i,g) = 1/T \max_{\mu_i\in\tilde{\Lambda}_i} \sum_{t=1}^T \varrho_t[i,g](\mu_i) - \varrho_t[i,g](\mu_i^t)$ the local regret associated with this local criterion. Remember that the projection can also be regarded as a weighted average or a conditional expectation. The last term corresponds to the future quadratic penalties the player will incur on the current scenario if policy μ_i is chosen and the other policy components are kept fixed. We also set $R_{loc}^{T,+}(i,g) = \max\left(R_{loc}^T(i,g), 0\right)$.

Progressive hiding is displayed in Algorithm 2.

Algorithm 2: Progressive Hiding

```
Initialization: Initialize RegrMin[i, g] for (i, g) ∈ I_p × G_i
for t ← 1 to T do
    foreach (i, g) ∈ I_p × G_i do
        μ_i^t(g) ← RegrMin[i, g].Decide()        // local no-regret decisions
    end
    γ ← Proj^t(μ_p^t)                            // Projection step
    foreach (i, g) ∈ I_p × G_i do
        θ_{i,g} ← (ϱ_t[i, g][δ_d])_{d∈[A_i(g)]}  // reward in the auxiliary game
        RegrMin[i, g].Observe(θ_{i,g})           // feed the local learners
    end
endfor
return γ
```

Interpretation of Algorithm 2. Suppose that at stage t of the algorithm, the policy is μ_i^t for $i \in I_p$. Then for $i \in I_p$ and $g \in G_i$, the local reward vector that is fed to the regret minimizer is $\mathbb{E}_{\delta_k,\mu_{-i}^t,\mu_{-p}^t}[r_p(\mathbf{h})|\tilde{\mathcal{X}}_i(\mathbf{h}) = g]$ with k in $[\mathcal{A}(g)]$. We add a term proportional to the gradient of $-||\gamma_i(g) - \mu_i^t(g)||^2$ so that we optimize for the penalized criterion (5). It should be noted that thanks to the projection step, the algorithm output an implementable policy no matter the choice of λ and $\tilde{\mathcal{X}}$. While the rational is different, we still note that the modification of a game payoff with a proximal term that makes the full criterion *policy-dependent* is also observed in [31]. In our case, however, the proximal term contains a projection, and is used to push the solution toward implementability.

5.2 Properties

Theorem 2. *Suppose $\tilde{\mathcal{X}}$ satisfies the perfect recall condition 2. Then the overall regret is bounded according to the relation $R^T \leq \sum_{i\in I_p}\sum_{g\in G_i} R_{loc}^{T,+}(i,g)$.*

Otherwise said, Theorem 2 states that CFR can be applied in the auxiliary game. However, note that the regret R^T is defined with respect to the auxiliary game payoff. Mirroring Theorem 1 for Algorithm 1, Theorem 2, with this bound on the regret in the auxiliary game, quantifies what Algorithm 2 seeks.

Proof Sketch for Theorem 2. The inductive proof borrows some aspects of [46] with the additional penalty terms that need to be accounted for.

The next property (which does not require perfect recall to hold) allows sizing the distance between μ^t and the implementable set.

Proposition 4.

$$1/T \sum_{t=1}^T \left(\lambda^t \mathbb{E}_{\mu_p^t,\mu_{-p}^t} \sum_{i\in I_p}\left[||\mu_i^t(\boldsymbol{h}) - Proj^t(\mu_p^t)_i(\boldsymbol{h})||^2\right]\right) \leq R^T + 2.||r||_\infty.$$

While we leave the choice of λ^t open, we report from our preliminary experiments that selecting a constant value with a grid search seems a viable strategy.

An important assumption to guarantee low-regret with counterfactual regret minimization is that the player (or the group of player) satisfies perfect recall. In practice, to cope with games of large size, the algorithm is often applied with an approximation of the game representation, which implies that perfect recall might not hold. Similarly, here, while we can provide guaranty when the relaxation induces perfect recall to hold, we also believe that for applications, an approximation of the perfect recall could also be envisioned for scalability concerns.

6 EXPERIMENTS

This paper focuses on games without perfect recall because we identify this setting as a relevant use case, since the method allows recovering Counterfactual Regret Minimization (CFR). In the competitive setting, a modified game can similarly be defined using either static information (hidden cards in the deck, dice results) or dynamic information (actions of the opponents) to improve the learning process, but this is left for future work.

This section reports on preliminary numerical experiments that complement the theoretical findings. We tested three implementations of Progressive Hiding on three different (team) games.

6.1 Trade Comm

We run two versions of progressive hiding on **Trade Comm** [20] because the game is elementary to solve for a human, but appears to be quite challenging for learning algorithms [40]. Furthermore, the game is parametrized by two parameters, which allows us to test different settings. The code is provided in the Supplementary Material. Of independent interest, the code closely aligns with the product game description, hence despite their equivalence in our context, the tree, and product representations lead to distinct coding paradigms. The no-regret learner we used for this game is FTRL with entropic regularization.

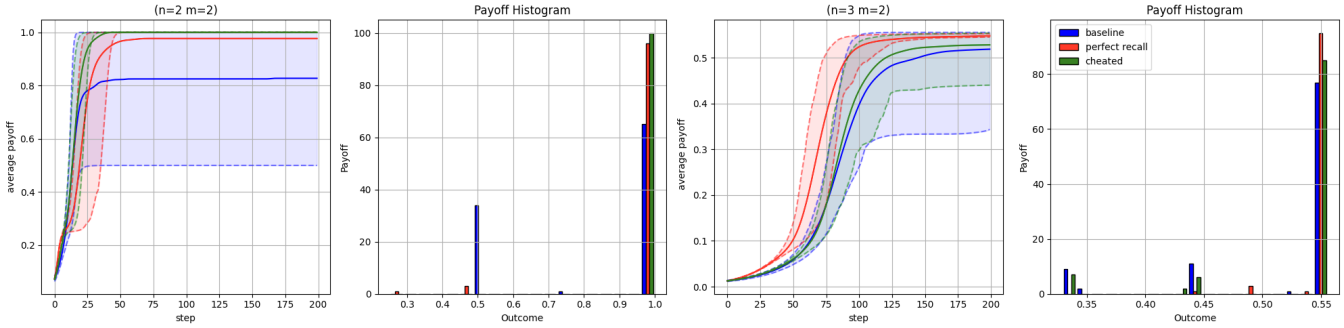

Figure 1: Learning outcomes ($\mathbb{E}_{\gamma^t}[r_t(\mathbf{h})]$) distribution for the three information map BASELINE, RECALL and CHEATED on Trade Comm with parameter $(m, n) = (2, 2)$ (left) and $(m, n) = (3, 2)$ (right).

Trade Comm is a common-payoff communication game. Each of the two players randomly receives one of several items in $[n]$ (s_1 and s_2). The first player communicates by choosing one out of several possible messages in $[m]$, denoted by m_1, which the second player observes. Then, the second player also selects a message (in $[m]$), denoted by m_2, for the first player to see. Both players then secretly request a trade involving one of the possible item combinations $((d_1^1, d_2^1)$ and $(d_1^2, d_2^2))$. A trade is successful if both players request to exchange their items for each other's. They both earn a point if the trade works out, and none if it doesn't. Despite seeming straightforward, Trade Comm effectively highlights the challenges in same payoff games. We showcase that information relaxation offers a range of strategies through the choice of the information map $\tilde{\mathcal{X}}$ and the penalty schedule λ^t.

Formally, $P = 1$, $(s_1, s_2) \in \Omega = [n] \times [n]$, $\mathbb{P}$ is the uniform distribution on Ω, the product space $\mathbb{H}$ is

$$\underbrace{([n] \times [n])}_{\Omega} \times \underbrace{[m]}_{\text{message 1}} \times \underbrace{[m]}_{\text{message 2}} \times \underbrace{([n] \times [n])}_{\text{trade request 1}} \times \underbrace{([n] \times [n])}_{\text{trade request 2}}.$$

The information map $\mathcal{X}$ is $\mathcal{X}_1(h) = s_1$, $\mathcal{X}_2(h) = s_2$, $\mathcal{X}_3(h) = (s_1, m_1, m_2)$, $\mathcal{X}_4(h) = (s_2, m_2, m_1)$, the available actions map $\mathcal{A}$ is $\mathcal{A}_1(h) = [m]$, $\mathcal{A}_2(h) = [m]$, $\mathcal{A}_3(h) = [n] \times [n]$, $\mathcal{A}_4(h) = [n] \times [n]$, and, last, the reward is $r_p(h) = [d_1^1 = s_1] \cdot [d_2^2 = s_1] \cdot [d_1^2 = s_2] \cdot [d_2^1 = s_2]$. We can check that perfect recall is not satisfied for $n > 1$.

We envision three possible information maps: the "original" information map $\mathcal{X}$, the "cheater" information map that reveals the private information, but does not record the message sent. This type of imperfect recall can be desired for computational efficiency in the same spirit as game abstractions are used: $\mathcal{X}_1^{cheat}(h) = (s_1, s_2)$, $\mathcal{X}_2^{cheat}(h) = (s_2, s_1)$, $\mathcal{X}_3^{cheat}(h) = (s_1, s_2, m_2)$, $\mathcal{X}_4^{cheat}(h) = (s_2, s_1, m_1)$, and a *perfect recall* information map, that, as it names suggest, ensures perfect recall: $\mathcal{X}_1^{PR}(h) = (s_1)$, $\mathcal{X}_2^{PR}(h) = (s_1, s_2, m_1)$, $\mathcal{X}_3^{PR}(h) = (s_1, s_2, m_1, m_2)$, $\mathcal{X}_4^{PR}(h) = (s_1, s_2, m_1, m_2, d_1^1, d_2^1)$. We insist that the information maps remain static during training. Progressive hiding is achieved entirely by adjusting the penalty parameter sequence. Both the cheater map and the perfect recall map could be used. The paper does not address which map is theoretically better, as this is beyond the scope of the current work. However, Theorem 2 suggests that ensuring perfect recall is a reasonable approach.

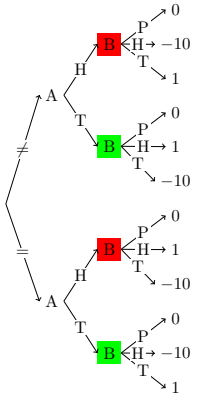

Figure 2: Tree representation of Cooperative Matching Pennies, introduced in Section 6.2. First a random state is sampled among SAME or DIFFERENT. Alice, the first player, observes the outcome of this random event and then chooses between TAIL and HEAD. Bob, the second player, knows Alice's choice but does not know the state of nature. Bob then makes his decision, choosing either TAIL, HEAD, or PASS. The payoff, indicated in the leaves, is the same for both player.

The Appendix in the supplementary material contains more details on the experiments. We compare the three information maps for $(n, m) = (2, 2)$ and $(n, m) = (3, 2)$ in Figure 1. The dashed lines correspond to the 10% quantiles, and the full line to the average value, over 100 runs with randomized initial policy. We see that the methods relying on progressive hiding beat the baseline on the two examples. Progressive hiding reaches the optimal payoff more often and faster than the baseline. It is notable that even *cheated*, which consists in an information relaxation not big enough to satisfy perfect recall, still induces a clear improvement over the baseline.

6.2 Cooperative Matching Pennies

The game is described in Figure 2. We designed this game so that it is a difficult game for the agents, but the solution is easy to see. Indeed, Bob is incentivized to pass in the early learning stages, so that the learning stops. We pitted a Monte-Carlo version of CFR versus its

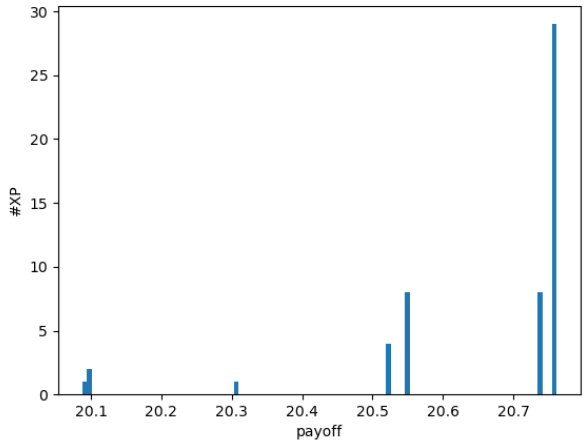

Figure 3: Distribution of the best maximal payoff obtained along the learning for each of the 50 training for Abstracted Tiny Bridge. According to [40], 20.32 is the performance of the best joint policy that does not requires coordination.

progressive hiding equivalent. We used a penalty parameter of 0.05 for progressive hiding. We considered the learning successful if, after 400 episodes, the algorithms reached an expected payoff greater than 0.95. We repeated the experiment 1000 times. While CFR never succeeded, progressive hiding succeeded 48% of the time.

6.3 Abstracted Tiny Bridge

This game is available in Open Spiel [20][8]. Abstracted Tiny Bridge is a simplified, cooperative version of contract bridge that preserves key strategic aspects. In this game, each player privately receives one of 12 possible hands. The players then engage in bidding to set the contract. The overall payoff is determined by the selected contract, the hand of the player who chose the contract, and the hand of the other player. The game's challenge lies in using bids to both communicate hand information and establish the contract, with fewer choices remaining as bidding progresses. We tested progressive hiding with regret matching. Instead of setting a value for the penalty parameter, we used a dynamic penalty parameter to control the expected payoff of the relaxed policy. The result of this experiment is summarized in Figure 3. It is notable that we were able to achieve those results using only 75 episodes, while the authors in [40] used 10 million episodes for their baselines, and 100 thousands for their algorithm (CAPI). We were unable to achieve the payoff of CAPI (close to 21, while our peak is at 20.76) though, still progressive hiding enjoys performance comparable to the other methods but with a microscopic budget of episodes.

7 DISCUSSION

In this article, we show how information relaxation can be made compatible with learning in games. We build on no-regret learners to propose progressive hiding. We show that the algorithm is principled and showcase promising experimental results.

Limitation. The main limitation of progressive hiding is its local nature, which stems from our definition of regret. Another limitation of progressive hiding is its scalability, but we can be optimistic in the possibility to combine the method with Monte Carlo sampling [21] and deep learning approaches [5, 13]. Last, a question that is not addressed in this work is how to choose λ^t.

Further work. A question that is not fully answered in this work is the choice of information map to perform the information relaxation. It seems intuitive that this question relates to the literature on game abstractions. Also, while we introduced information maps (*i.e.*, functions) to encode the player's knowledge for the sake of simplicity, it seems that information fields [9] because they have a lattice and algebraic structure, would be a better mathematical tool if one wants to go further. Typically, an information field of interest is the smallest field greater than the original field that ensures that perfect recall is satisfied. In this direction, it would be interesting to know if there exist informational properties other than perfect recall that allow for decomposition results.

[8]see https://github.com/google-deepmind/open_spiel/blob/master/open_spiel/games/tiny_bridge/tiny_bridge.h for a brief explanation by the developers of the library

ACKNOWLEDGMENTS

BH would like to thank Michel De Lara for introducing him to Witsenhausen's model. We thank the anonymous reviewers and the area chair for their constructive feedback.

REFERENCES

[1] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. 2020. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence* 280 (2020), 103216. https://doi.org/10.1016/j.artint.2019.103216
[2] Avrim Blum and Yishay Mansour. 2007. *Learning, Regret Minimization, and Equilibria*. Cambridge University Press, 79–102.
[3] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2015. Heads-up limit hold'em poker is solved. *Science* 347, 6218 (2015), 145–149.
[4] David B Brown, James E Smith, et al. 2022. Information relaxations and duality in stochastic dynamic programs: A review and tutorial. *Foundations and Trends® in Optimization* 5, 3 (2022), 246–339.
[5] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. 2019. Deep Counterfactual Regret Minimization. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 793–802. https://proceedings.mlr.press/v97/brown19b.html
[6] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.
[7] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365, 6456 (2019), 885–890.
[8] Michael Buro, Jeffrey Richard Long, Timothy Furtak, and Nathan Sturtevant. 2009. Improving state evaluation, inference, and search in trick-based card games. In *Twenty-First International Joint Conference on Artificial Intelligence*.
[9] Pierre Carpentier, Jean-Philippe Chancelier, Guy Cohen, and Michel De Lara. 2015. Stochastic multi-stage optimization. *Probability Theory and Stochastic Modelling* 75 (2015).
[10] Ian Frank and David Basin. 1998. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence* 100, 1-2 (1998), 87–123.
[11] Timothy Furtak and Michael Buro. 2013. Recursive Monte Carlo search for imperfect information games. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. IEEE, 1–8.
[12] Matthew L Ginsberg. 2001. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research* 14 (2001), 303–358.
[13] Daniel Hennes, Dustin Morrill, Shayegan Omidshafiei, Rémi Munos, Julien Perolat, Marc Lanctot, Audrunas Gruslys, Jean-Baptiste Lespiau, Paavo Parmas, Edgar Duéñez-Guzmán, et al. 2020. Neural replicator dynamics: Multiagent learning via hedging policy gradients. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems*. 492–501.

[14] Benjamin Heymann, Michel De Lara, and Jean-Philippe Chancelier. 2021. Causal inference theory with information dependency models. *arXiv preprint arXiv:2108.03099* (2021).

[15] Benjamin Heymann, Michel De Lara, and Jean-Philippe Chancelier. 2022. Kuhn's equivalence theorem for games in product form. *Games and Economic Behavior* 135 (2022), 220–240.

[16] Hengyuan Hu and Jakob N Foerster. 2019. Simplified action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1912.02288* (2019).

[17] Michael Johanson, Nolan Bard, Marc Lanctot, Richard G Gibson, and Michael Bowling. 2012. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization.. In *Aamas*. 837–846.

[18] Harold W Kuhn. 1953. Extensive games and the problem of information. *Contributions to the Theory of Games* 2, 28 (1953), 193–216.

[19] Marc Lanctot, Richard Gibson, Neil Burch, Martin Zinkevich, and Michael Bowling. 2012. No-regret learning in extensive-form games with imperfect recall. *arXiv preprint arXiv:1205.0622* (2012).

[20] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. 2019. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453* (2019).

[21] 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*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/00411460f7c92d2124a67ea0f4cb5f85-Paper.pdf

[22] Na Li, Jason R Marden, and Jeff S Shamma. 2009. Learning approaches to the Witsenhausen counterexample from a view of potential games. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 157–162.

[23] Viliam Lisỳ, Marc Lanctot, and Michael H Bowling. 2015. Online Monte Carlo Counterfactual Regret Minimization for Search in Imperfect Information Games.. In *AAMAS*. 27–36.

[24] Jeffrey Long, Nathan Sturtevant, Michael Buro, and Timothy Furtak. 2010. Understanding the success of perfect information monte carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. 134–140.

[25] J. McCarthy. 1990. Chess as the Drosophila of AI. In *Computers, Chess, and Cognition*, T. Anthony Marsland and Jonathan Schaeffer (Eds.). Springer New York, New York, NY, 227–237.

[26] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, 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.

[27] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. 2013. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Trans. Automat. Control* 58, 7 (2013), 1644–1658.

[28] Martin J. Osborne and Ariel Rubinstein. 1994. *A course in game theory*. The MIT Press, Cambridge, USA. electronic edition.

[29] Neal Parikh, Stephen Boyd, et al. 2014. Proximal algorithms. *Foundations and trends® in Optimization* 1, 3 (2014), 127–239.

[30] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. 2022. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (2022), 990–996.

[31] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, et al. 2021. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *International Conference on Machine Learning*. PMLR, 8525–8535.

[32] Klaus Ritzberger et al. 2016. *The theory of extensive form games*. Springer.

[33] R. Tyrrell Rockafellar. 1970. *Convex Analysis*. Princeton University Press.

[34] R. Tyrrell Rockafellar and Jie Sun. 2019. Solving monotone stochastic variational inequalities and complementarity problems by progressive hedging. *Mathematical Programming* 174, 1 (2019), 453–471.

[35] R. Tyrrell Rockafellar and Roger J-B Wets. 1991. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research* 16, 1 (1991), 119–147.

[36] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

[37] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. 2021. *Lectures on stochastic programming: modeling and theory*. SIAM.

[38] Y. Shoham and K. Leyton-Brown. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

[39] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[40] Samuel Sokota, Edward Lockhart, Finbarr Timbers, Elnaz Davoodi, Ryan D'Orazio, Neil Burch, Martin Schmid, Michael Bowling, and Marc Lanctot. 2021. Solving Common-Payoff Games with Approximate Policy Iteration. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 11 (May 2021), 9695–9703. https://doi.org/10.1609/aaai.v35i11.17166

[41] Christopher Solinas, Douglas Rebstock, and Michael Buro. 2019. Improving search with supervised learning in trick-based card games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1158–1165.

[42] Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. 2012. Bayes' bluff: Opponent modelling in poker. *arXiv preprint arXiv:1207.1411* (2012).

[43] Nathan Sturtevant. 2008. An analysis of UCT in multi-player games. *ICGA Journal* 31, 4 (2008), 195–208.

[44] Hans S. Witsenhausen. 1968. A counterexample in stochastic optimum control. *SIAM Journal on Control* 6, 1 (1968), 131–147.

[45] Hans S. Witsenhausen. 1971. On Information Structures, Feedback and Causality. *SIAM Journal on Control* 9, 2 (1971), 149–160. https://doi.org/10.1137/0309013 arXiv:https://doi.org/10.1137/0309013

[46] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. 2007. Regret minimization in games with incomplete information. *Advances in neural information processing systems* 20 (2007).

[47] Martin Zinkevirch. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*. 928–936.