Model-Free Online Learning in Unknown Sequential Decision Making Problems and Games

Abstract

Regret minimization has proved to be a versatile tool for sequential decision making and extensive-form games. In large two-player zero-sum imperfect-information games, modern extensions of counterfactual regret minimization (CFR) are currently the practical state of the art for computing a Nash equilibrium. Most regret-minimization algorithms for sequential decision making, including CFR, require (i) an exact *model* of the player's decision nodes, observation nodes, and how they are linked, and (ii) full knowledge, at all times t, about the payoffs—even in parts of the decision space that are not encountered at time t. Recently, there has been growing interest towards relaxing some of those restrictions and making regret minimization applicable to settings for which reinforcement learning methods have traditionally been used—for example, those in which only black-box access to the environment is available. We give the first, to our knowledge, regret-minimization algorithm that guarantees sublinear regret with high probability even when requirement (i)—and thus also (ii)—is dropped. We formalize an online learning setting in which the strategy space is not known to the agent and gets revealed incrementally whenever the agent encounters new decision points. We give an efficient algorithm that achieves $O(T^{3/4})$ regret with high probability for that setting, even when the agent faces an adversarial environment. Our experiments show it significantly outperforms the prior algorithms for the problem, which do not have such guarantees. It can be used in any application for which regret minimization is useful: approximating Nash equilibrium or quantal response equilibrium, approximating coarse correlated equilibrium in multi-player games, learning a best response, learning safe opponent exploitation, and online play against an unknown opponent/environment.

1 Introduction

A *sequential decision making (SDM)* problem formalizes in a tree-form structure the interaction of an agent with an unknown and potentially adversarial environment. The agent's tree includes decision nodes, observation nodes, and terminal nodes. SDM captures the problem that a player faces in an extensive-form game. SDM also captures MDPs and POMDPs where the agent conditions on observed history, but SDM problems are more general because the Markovian assumption is not necessarily made.

In SDM, the environment may react adversarially to what the agent does. This is important to take into account, for example, in game-theoretic settings and *multiagent reinforcement learning (MARL)* because the other agents' learning makes the environment nonstationary for the agent (Sandholm and Crites 1996; Matignon, Laurent, and Le Fort-Piat 2012). This is in contrast to the standard assumption in single-agent reinforcement learning where the environment is *oblivious* to the agent instead of adversarial. Hence, learning strong *policies* (aka. *strategies*) in SDM problems is especially challenging, and the agent must be careful about exploration because exploration actions can change the environment.

Online, regret minimization methods have been successfully used in SDM. In particular, the *counterfactual regret minimization (CFR)* framework decomposes the overall regret of an agent to local regrets at individual decision nodes (aka. information sets in game theory) (Zinkevich et al. 2007). That enables significantly larger SDMs to be tackled. Many enhancements have been developed on top of the basic CFR framework (Lanctot et al. 2009; Tammelin 2014; Brown and Sandholm 2015; Brown, Kroer, and Sandholm 2017; Brown and Sandholm 2017a, 2019a; Farina, Kroer, and Sandholm 2019), and have led to major milestones in imperfect-information games such as poker (Bowling et al. 2015; Moravčík et al. 2017; Brown and Sandholm 2017b, 2019b). Many of those methods *guarantee* low regret even against an adversarial environment—which, in turn, enables the computation of game-theoretic solutions such as Nash equilibrium, coarse correlated equilibrium (Moulin and Vial 1978; Celli et al. 2020), best responses, etc.

However, those methods usually come with two drawbacks: (i) they require an explicit upfront *model* of the agent's decision space, and (ii) depending on the online learning model used, they require full feedback, at all times t, about the payoffs assigned by the environment—even in parts of the decision space not encountered at time t. There has been work towards an online learning setting, called the *bandit optimization setting*, that drops (ii) (Lattimore and Szepesvári 2020). Most MARL algorithms apply for the unknown game setting and drop both (i) and (ii), often at the cost of invalidating any regret guarantees. In this paper, we give, to our knowledge, the first regret-minimization algorithm that guarantees sublinear (specifically $O(T^{3/4}\sqrt{\log 1/p})$ with probability $1-p$) regret even when requirements (i) and (ii) are both dropped.

Conceptually, our algorithm has elements of both online bandit optimization and MARL. On the one hand, our regret guarantees hold with high probability no matter how the environment picks its actions or assigns utilities to terminal nodes at all iterations t. In fact, our algorithm is a regret minimizer in the proper online-learning sense: it does not need to know—and makes no assumptions about—the underlying policy of the environment or the utility at the terminal states. Furthermore, those quantities are not assumed to be time independent and they can even be selected adversarially by the environment at each iteration. This is contrast with many *self-play* methods, that require control over the opponent in order to retain guarantees. In particular, because we assume no control over the adversarial environment, every interaction with the environment can lead it to react and change behavior in the next iteration. So, it is impossible to "freeze" the policy of the environment to perform off-policy exploration like many self-play methods require. Because of its strong online guarantees, our algorithm can be used for all the applications in which regret minimization provides benefits—for example, to converge to Nash equilibrium in a two-player zero-sum extensive-form game, to learn a best response against static opponents, to converge to coarse correlated equilibrium in a multiagent setting (Moulin and Vial 1978; Celli et al. 2020), to converge to a quantal-response equilibrium (Ling, Fang, and Kolter 2018; Farina, Kroer, and Sandholm 2019), to compute safe exploitative strategies (Farina, Kroer, and Sandholm 2019; Ponsen, De Jong, and Lanctot 2011), or to play online against an unknown opponent/environment.

On the other hand, ours is *not* an algorithm for the online bandit optimization problem. In bandit optimization, the agent does *not* interact with the environment: at all times t, the agent outputs a policy $\boldsymbol{\pi}^t$ for the whole decision space, and receives a single real number u^t, representing the gain incurred by $\boldsymbol{\pi}^t$, as feedback. Instead, our algorithm operates within a slightly different online learning model that we introduce, which we call the *interactive bandit model*. In it, the decision maker gets to observe the signals (actions) selected by the environment on the path from the root of the decision problem to the agent's terminal state, in additions to u^t. Hence, we operate within a more specific online learning model than bandit optimization (which applies to, for example, picking a point on a sphere also, as opposed to just SDM), and rather one that is natural to SDM. This comes with a significant advantage. While, to our knowledge, all algorithms for the bandit optimization problem require *a priori* knowledge of the full structure of the sequential decision problem,[1] our algorithm for the interactive bandits model is *model free*. Here, the structure of the sequential decision process is at first unknown and has to be discovered by exploring as part of the decision-making process. Decision and observation nodes are revealed only at the time the agent encounters them for the first time.

[1]This is needed, for example, to construct a self-concordant barrier function (Abernethy, Hazan, and Rakhlin 2008) or a global regularizer for the strategy space (Abernethy and Rakhlin 2009; Farina, Schmucker, and Sandholm 2020), to compute a barycentric spanner (Awerbuch and Kleinberg 2004; Bartlett et al. 2008), or to set up a kernel function (Bubeck, Lee, and Eldan 2017).

Closely Related Research

In the rest of this section we discuss how our algorithm relates to other attempts to connect online learning guarantees with model-free MARL. A comparison between our algorithm and other settings in online learning is deferred to Section 3, where we formally introduce our interactive bandit model.

The greatest inspiration for our algorithm is the *online variant of Monte Carlo CFR (online MCCFR)* algorithm proposed in passing by (Lanctot et al. 2009). Unlike the traditional "self-play" MCCFR, *online* MCCFR does not assume that the algorithm have control over the environment. The authors note that in theory online MCCFR could be used to play games in a model-free fashion, provided that a lower bound on the reach probability of every terminal state can be enforced. That lower bound, say η, is necessary for them to invoke their main theorem, which guarantees $O(\eta^{-1}T^{1/2}\sqrt{1/p})$ regret with probability $1-p$. They suggest perhaps using some form of ϵ-greedy exploration at each decision node to enforce the lower bound, but no guarantee is provided and the authors then move away from this side note to focus on the self-play case. We show that their proposed approach encounters significant hurdles. First, using exploration at each decision node results in a lower bound on the reach of every terminal state on the order of $\eta = \epsilon^d$, where d is the depth of the decision process, thus making the regret bound not polynomial in the size of the decision process itself, but rather exponential. Second, the paper did not provide theoretical guarantees for the online case. In particular, the theory does not take into account the degredation effect caused by the exploration itself, which scales roughly as ηT. So, on the one hand, a large η is needed to keep the term η^{-1} in their regret bound under control, but at the same time a large η results in a ηT term being added to the regret. These hurdles show that it is unlikely that their approach can lead to $O(T^{1/2}\sqrt{1/p})$ regret with high probability $1-p$ as they hypothesized. We address those issues by using a different type of exploration and being careful about bounding the degradation terms in the regret incurred due to the exploration. Because of the latter, our algorithm incurs $O(T^{3/4}\sqrt{\log(1/p)})$ regret with high probability against adversarial opponents. Because the exponent is less than 1, ours is truly a regret minimization algorithm for SDMs with unknown structure, and to our knowledge, the first. At the same time, our exponent is worse than the originally hypothesized exponent 1/2. It is unknown whether the latter can be achieved. Finally, our dependence on p is better than in their hypothesized regret bound.

A recent paper by Srinivasan et al. (2018) related policy gradient algorithms to CFR (and, to a much lesser degree, MCCFR). Despite their experiments demonstrating empirical convergence rates for sampled versions of their algorithms, formal guarantees are only obtained for tabularized policy iteration in self-play, and use policy representations that require costly ℓ_2 projections back into the policy space. In contrast, our regret guarantees hold (i) in any SDM setting (not just two-player zero-sum extensive-form games), (ii) in high probability, (iii) with sampling, and (iv) when playing against any environment, even an adversarial one, without

requiring complex projections.

A very recent paper (Zhou, Li, and Zhu 2020) on two-player zero-sum games proposes combining full-information regret minimization with posterior sampling (Strens 2000) to estimate the utility function of the player and transition model of chance, both of which they assume to be time independent unlike our setting. They show in-expectation bounds under the assumption that the agent's strategy space is known *ex ante*. We operate in a significantly more general setting where the observations and utilities are decided by the environment and can change—even adversarially—between iterations. Like theirs, our algorithm converges to Nash equilibrium in two-player zero-sum games when used in self play. However, unlike theirs, our algorithm is a regret minimizer that can be used for other purposes also, such as finding a coarse correlated equilibrium in multiplayer general-sum games, a quantal-response equilibrium, or safe exploitative strategies. In the latter two applications, the payoff function, in effect, changes as the agent changes its strategy. Our regret guarantees hold in high probability and we do not assume *ex ante* knowledge of the agent's strategy space.

A different line of work has combined fictitious play (Brown 1951) with deep learning for function approximation (Heinrich, Lanctot, and Silver 2015; Heinrich and Silver 2016). Those methods do not give regret guarantees. Finally, other work has studied how to combine the guarantees of online learning with MDPs. Kash, Sullins, and Hofmann (2020) combine the idea of breaking up and minimizing regret locally at each decision point, proper of CFR, with Q-learning, obtaining an algorithm with certain in-the-limit guarantees for MDPs. Even-Dar, Kakade, and Mansour (2009) study online optimization (in the full-feedback setting, as opposed to bandit) in general MDPs where the reward function and the structure of the MDP is known. Neu et al. (2010) study online bandit optimization in MDPs in the *oblivious* setting, achieving $O(T^{2/3})$ regret with high probability, again assuming that the MDP's structure is known and certain conditions are met. Zimin and Neu (2013) give bandit guarantees for episodic MDPs with a fixed known transition function.

2 Our Model for (Unknown) Sequential Decision Making and Games

In this section, we introduce the notation for SDM problems that we will be using the rest of the paper.

A *sequential decision making (SDM) problem* is structured as a tree made of three types of nodes: (i) *decision nodes* j, in which the agent acts by selecting an action from the finite set A_j (different decision nodes can admit different sets of allowed actions); (ii) *observation points* k, where the agent observes one out of set S_k of finitely many possible signal from the environment (different observation points can admit different sets of possible signals); and (iii) *terminal nodes*, corresponding to the end of the decision process. We denote the set of decision nodes in the sequential decision making problem as $\mathcal{J}$, the set of observation points as $\mathcal{K}$, and the set of terminal nodes as Z. Furthermore, we let ρ denote the dynamics of the game: selecting action $a \in A_j$ at decision node $j \in \mathcal{J}$ makes the process transition to the next state $\rho(j, a) \in \mathcal{J} \cup \mathcal{K} \cup Z$, while the process transitions to $\rho(k, s) \in \mathcal{J} \cup \mathcal{K} \cup Z$ whenever the agent observes signal $s \in S_k$ at observation point $k \in \mathcal{K}$.

Our algorithm operates in the difficult setting where the structure of the SDM problem is at first unknown and can be discovered only through exploration. Decisions and observation nodes are revealed only at the time the agent encounters them for the first time. As soon as a decision node j is revealed for the first time, its corresponding set of actions A_j is revealed with it.

Sequences In line with the game theory literature, we call a *sequence* a decision node-action pair; each sequence (j, a) uniquely identifies a path from the root of the decision process down to action a at decision node j, included. Formally, we define the set of sequences as $\Sigma := \{(j, a) : j \in \mathcal{J}, a \in A_j\} \cup \{\varnothing\}$, where the special element $\varnothing$ is called the *empty sequence*. Given a decision node $j \in \mathcal{J}$, its *parent sequence*, denoted p_j, is the last sequence (that is, decision node-action pair) encountered on the path from the root of the decision process down to j. If the agent does not act before j (that is, only observation points are encountered on the path from the root to j), we let $p_j = \varnothing$.

Given a terminal node $z \in Z$ and a sequence $(j, a) \in \Sigma$, we write $(j, a) \rightsquigarrow z$ to mean that the path from the root of the decision process to z passes through action a at decision node j. Similarly, given a terminal node $z \in Z$ and an observation node-signal pair (k, s) ($s \in S_k$), we write $(k, s) \rightsquigarrow z$ to indicate that the path from the root of the decision process to z passes through signal s at observation node k. Finally, we let $\sigma(z)$ be the last sequence (decision node-action pair) on the path from the root of the decision process to terminal node $z \in Z$.

Strategies Conceptually, a strategy for an agent in a sequential decision process specifies a distribution $\boldsymbol{x}_j \in \Delta^{|A_j|}$ over the set of actions A_j at each decision node $j \in \mathcal{J}$. We represent a strategy using the *sequence-form representation*, that is, as a vector $\boldsymbol{q} \in \mathbb{R}^{|\Sigma|}_{\geq 0}$ whose entries are indexed by Σ. The entry $q[j, a]$ contains the product of the probabilities of all actions at all decision nodes on the path from the root of the process down to and including action a at decision node $j \in \mathcal{J}$. A vector $\boldsymbol{q} \in \mathbb{R}^{|\Sigma|}_{\geq 0}$ is a valid sequence-form strategy if and only if it satisfies the constraints (i) $\sum_{a \in A_j} q[j, a] = q[p_j]$ for all $j \in \mathcal{J}$; and (ii) $x[\varnothing] = 1$ (Romanovskii 1962; Koller, Megiddo, and von Stengel 1994; von Stengel 1996). We let Q denote the set of valid sequence-form strategies. Finally, we let $\Pi \subseteq Q$ denote the subset of sequence-form strategies whose entries are only 0 or 1; a strategy $\boldsymbol{\pi} \in \Pi$ is called a *pure* sequence-form strategy, as it assigns probability 1 to exactly one action at each decision node.

3 Online Learning and Our Interactive Bandit Model

In online learning, an agent interacts with its environment in this order: (i) The environment chooses a (secret) gradient vector $\boldsymbol{\ell}^t$ of bounded norm; (ii) The agent picks a pure strategy $\boldsymbol{\pi}^t \in \Pi$. The environment evaluates the reward

(gain) of the agent as $(\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t \in \mathbb{R}$; (iii) The agent observes some feedback about her reward. The feedback is used by the agent to learn to output good strategies over time. The learning is online in the sense that the strategy $\boldsymbol{\pi}^t$ at time t is output before any feedback for it (or future strategies) is available. A standard quality metric for evaluating an agent in this setting is the *regret* that she accumulates over time: $R^T(\hat{\boldsymbol{\pi}}) := \sum_{t=1}^T (\boldsymbol{\ell}^t)^\top \hat{\boldsymbol{\pi}} - \sum_{t=1}^T (\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t$. This measures the difference between the total reward accumulated up to time T, compared to the reward that would have been accumulated had the oracle output the fixed strategy $\hat{\boldsymbol{\pi}} \in \Pi$ at all times. A "good" agent, that is, a *regret minimizer*, is one whose regret grows sublinearly: $R^T(\hat{\boldsymbol{\pi}}) = o(T)$ for all $\hat{\boldsymbol{\pi}} \in \Pi$.

Online learning models vary based on the type and extent of feedback that is made available to the agent. We will focus on two existing models—namely, the full-feedback[2] setting and the bandit linear optimization setting. Then we will introduce a third model that is especially natural for SDM.

Full-Feedback[2] Setting Here, the environment always reveals the full gradient vector $\boldsymbol{\ell}^t$ to the agent (after the strategy has been output). This is the same setting that was proposed in the landmark paper by Zinkevich (2003) and is the most well-studied online optimization setting. Efficient agents that guarantee $O(T^{1/2})$ regret with high probability in the full-feedback setting are known well beyond SDM and extensive-form games (e.g, (Shalev-Shwartz 2012; Hazan 2016)). In fact, given any convex and compact set $\mathcal{X}$, it is possible to construct an agent that outputs decisions $\boldsymbol{x}^t \in \mathcal{X}$ that achieves $O(T^{1/2})$ regret *in the worst case*, even when the gradient vectors $\boldsymbol{\ell}^t$ are chosen adversarially by the environment *after* the decision $\boldsymbol{x}^t$ has been revealed. In the specific context of SDM and extensive-form games, the most widely-used oracle in the full-feedback setting is based on the counterfactual regret (CFR) framework (Zinkevich et al. 2007). The idea is to decompose the task of computing a strategy for the whole decision process into smaller subproblems at each decision point. The local strategies are then computed via $|\mathcal{J}|$ independent full-information regret minimizers, one per decision node, that at each t observe a specific feedback that guarantees low global regret across the decision process. CFR guarantees $O(T^{1/2})$ regret with high probability against any strategy $\boldsymbol{\pi} \in \Pi$.

Bandit Linear Optimization Here, the only feedback that the environment reveals to the agent is the utility $(\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t$ at each time t (Kleinberg 2004; Flaxman, Kalai, and McMahan 2005). Despite this extremely limited feedback, $\tilde{O}(T^{1/2})$ regret can still be guaranteed with high probability in some domains (including simplexes (Auer et al. 2002) and spheres (Abernethy and Rakhlin 2009)), although, to our knowledge, a polynomial algorithm that guarantees $\tilde{O}(T^{1/2})$ regret with high probability for any convex and compact domain has not been discovered yet.[3] Guaranteeing $\tilde{O}(T^{1/2})$ *in expectation* is possible for any domain of decisions (Abernethy, Hazan, and Rakhlin 2008), but unfortunately in-expectation low regret is not strong enough a guarantee to enable, for instance, convergence to Nash or correlated equilibrium as described above. In the specific case of sequential decision processes, (Farina, Schmucker, and Sandholm 2020) proposed a bandit regret minimizer that achieves $O(T^{1/2})$ regret in expectation compared to any policy and linear-time iterations. Upgrading to in-high-probability $O(T^{1/2})$ regret guarantees while retaining linear-time iterations remains an open problem.

Interactive Bandit We propose *interactive bandits* as a natural online learning model to capture the essence of sequential decision processes. Here, an agent interacts with the environment until a terminal state is reached, at which point the payoff (a real number) is revealed to the agent. The agent observes the environment's action (signal) whenever the interaction moves to an observation point. We formalize this as an online learning model as follows. At all times t, before the agent acts, the environment privately selects (i) a choice of payoff $u^t : Z \to \mathbb{R}$ for each terminal state $z \in Z$, and (ii) a secret choice of signals $s_k^t \in S_k$ for all observation points $k \in \mathcal{K}$. These choices are hidden, and only the signals relevant to the observation points reached during the interaction will be revealed. Similarly, only the payoff $u^t(z^t)$ relative to the terminal state z^t reached in the interaction will be revealed. In other words, the feedback that is revealed to the agent after the interaction is the terminal state z^t that is reached (which directly captures all signals revealed by the environment, as they are the signals encountered on the path from the root of the decision process to z^t) and its corresponding payoff $u^t(z^t)$, which can be equivalently expressed as $u^t(z^t) = (\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t$, where the gradient vector $\boldsymbol{\ell}^t$ is defined as the (unique) vector such that for all strategies $\boldsymbol{x} \in Q$,

$$(\boldsymbol{\ell}^t)^\top \boldsymbol{x} = \sum_{z \in Z} u^t(z) \left(\prod_{(k,s) \rightsquigarrow z} \mathbb{1}[s_k^t = s] \right) x[\sigma(z)]. \quad (1)$$

We assume that the environment is adversarial, in the sense that the environment's choices of payoffs u^t and signals s_k^t at time t can depend on the previous actions of the agent.

Our term "interactive bandits" comes from the fact that an algorithm for this setting can be thought of as interacting with the environment until a terminal state of the decision process is reached and a corresponding payoff is revealed. In other words, while for modeling purposes it is convenient to think about online learning algorithms as outputting strategies $\boldsymbol{\pi}$ for the *whole* strategy space, one can think of an interactive bandits algorithm as one that instead only outputs one action

[2] In online learning, "full-feedback" is typically called "full-information". We use "full-feedback" to avoid confusion with full-information games, that is, games where the full state is available to all players at all times.

[3] Several algorithms are able to guarantee one or two out of the three requirements (i) applicable to any convex domain, (ii) polynomial time per iteration, (iii) $\tilde{O}(T^{1/2})$ regret with high probability. For example, Bartlett et al. (2008) achieve (i) and (iii) by extending an earlier paper by Dani, Kakade, and Hayes (2008), and (György et al. 2007) achieves (ii) for the set of flows with suboptimal regret guarantees.

at a time as the interaction moves throughout the decision process.

Since the interactive bandit model requires that the gradient vector $\boldsymbol{\ell}^t$ be expressible as in (1), it makes more assumptions on the gradient vector than either the bandit linear optimization model or the full-feedback model. In terms of feedback, it is an intermediate model: it receives a superset of the feedback that the bandit linear optimization framework receives, but significantly less than the full-feedback model. So, theoretically, one could always use an algorithm for the bandit linear optimization model to solve a problem in the interactive bandit model. However, as we show in this paper, one can design a regret-minimization algorithm for the interactive bandit model that achieves sublinear $O(T^{3/4})$ regret with high probability *even in decision processes and extensive-form games whose decision space is at first unknown*. To our knowledge, no algorithm guaranteeing sublinear regret when the decision space is at first unknown has been designed for bandit linear optimization.

4 Algorithm for Unknown Sequential Decision Making Problems

We now describe a regret minimizer for the interactive bandit model. At all time t, it goes through two phases: first, the *rollout* phase, and then the *regret update* phase. During the rollout phase, the algorithm plays until a terminal state z^t and its corresponding payoff u^t are revealed. During the regret update phase, the algorithm rewinds through the decision nodes j encountered during that trajectory, and updates certain parameters at those decision nodes based on the newly-observed payoff u^t.[4] Like the CFR framework, our algorithm picks actions at each decision node j by means of a *local* full-information regret minimizer $\mathcal{R}_j$ for the action set A_j at that decision node.

Rollout Phase: Playing the Game

We describe two alternative algorithms for the rollout phase (namely the "upfont-flipping" and the "on-path-flipping" rollout variants), which differ in the way the mix exploration and exploitation. Both variants are theoretically sound, and yield to the same sublinear in-high-probability regret bound (Section 5), even when different variants are used at different times t while playing against the adversarial opponent.

We start from the "upfront-flipping" variant, which is arguably the conceptually simpler variant, although we find it to usually perform worse in practice.

Upfront-Flipping Rollout Variant When the *upfront-flipping rollout variant* is used at time t, at the beginning of the rollout phase and before any action is played, a biased coin is tossed to decide the algorithm to use to play out the interaction:

- With probability β^t, the EXPLORE routine is used. It ignores the recommendations of the local regret minimizers at each decision node and instead plays according to the *exploration function* $h^t : \Sigma \to \mathbb{R}_{>0}$ (more details are below). In particular, at every decision node j encountered during the rollout, the agent picks action $a \in A_j$ at random according to the distribution $h^t(j,a)/(\sum_{a' \in A_j} h^t(j,a'))$.
- With probability $1-\beta^t$, the EXPLOIT routine is used. With this routine, at every decision node j encountered during the rollout, the agent picks a decision by sampling from the distribution $\boldsymbol{x}_j^t \in \Delta^{|A_j|}$ recommended by the regret minimizer $\mathcal{R}_j$.

In both cases, a regret minimizer $\mathcal{R}_j$ for decision node j is created when j is first discovered.

Our upfront-flipping rollout strategy differs from the ϵ-greedy strategy in that the coin is tossed for the entire trajectory, not at each decision point.

On-Path-Flipping Rollout Variant In the *on-path-flipping* rollout variant, there is no single coin toss to distinguish between exploration and exploitation, and the two are interleaved throughout the rollout. Before any action is picked, the two *reach* quantities $r^t, \hat{r}^t$ are both set to 1. Then, the rollout phase begins, and eventually the agent will be required to make a decision (pick an action) at some decision point j. Let $\boldsymbol{x}_j^t \in \Delta^{|A_j|}$ be the distribution over actions recommended by the regret minimizer $\mathcal{R}_j$. In the on-path-flipping rollout variant, the agent picks an action $a \in A_j$ at j with probability proportional to

$$(1-\beta^t) r^t \cdot x_j^t[a] + \beta^t \hat{r}^t \cdot \frac{h^t(j,a)}{\sum_{a' \in A_j} h^t(j,a')}.$$

Let a^* be the chosen action; r^t and $\hat{r}^t$ are updated according to the formulas $r^t \leftarrow r^t \cdot x_j^t[a^*]$ and $\hat{r}^t \leftarrow \hat{r}^t \cdot \frac{h^t(j,a^*)}{\sum_{a' \in A_j} h(j,a')}$. The agent keeps using this specific way of selecting actions and updating the reach quantities $r, \hat{r}$ for all decision points encountered during the rollout.

The role of h^t In both variants, the role of the exploration function h^t is to guide exploration of different parts of the decision process.[5] The optimal choice for h^t is to have $h^t(j,a)$ measure the number of terminal states in the subtree rooted at (j,a). When this information is not available, a heuristic can be used instead. If no sensible heuristic can be devised, the uniform exploration strategy $h^t(j,a) = 1$ for all $(j,a) \in \Sigma$ is always a valid fallback. In Theorem 1 below, we give guarantees about the regret cumulated by our algorithm that apply to any $h^t : \Sigma \to \mathbb{R}_{>0}$.

Regret Update Phase: Propagating the Payoff up the Tree

In the regret update phase, the revealed feedback (that is, the revealed utility $u^t(z^t)$ and the terminal state z^t that was

[4] While our algorithm shares many of the building blocks on Monte Carlo Tree Search (MCTS)—incremental tree-building, backpropagation, rollouts—it is *not* an anytime search algorithm, at least not in the sense of traditional game-tree search like the one employed by the Online Outcome Sampling algorithm (Lisý, Lanctot, and Bowling 2015).

[5] Despite the positive exploration term induced by h^t, it is not guaranteed that all decision points will be discovered as $T \to \infty$ as the adversarial environment might prevent so. Nonetheless, our algorithm guarantees sublinear regret with high probability.

reached in the rollout phase) is used to construct suitable *local* gradient vectors $\boldsymbol{\ell}_j^t$ for each of the local regret minimizers $\mathcal{R}_j$ on the path from the terminal state z^t up to the root. Let $(j_1, a_1), \dots, (j_m, a_m)$ be the sequence of decision nodes and actions that were played, in order, that ultimately led to terminal state z^t during the repetition of the game at time t. We start to construct local gradient vectors from decision node j_m, where we set the three quantities

$$\gamma^t := (1-\beta^t) \cdot \prod_{i=1}^{m} x_{j_i}^t[a_i] + \beta^t \cdot \prod_{i=1}^{m} \frac{h^t(j_i, a_i)}{\sum_{a' \in A_{j_i}} h^t(j_i, a')},$$

$$\hat{u}_{j_m}^t := \frac{u^t(z^t)}{\gamma^t}, \quad \boldsymbol{\ell}_{j_m}^t := \hat{u}_{j_m}^t \boldsymbol{e}_{a_m},$$

where we used the notation $\boldsymbol{e}_{a_m} \in \Delta^{A_j}$ to denote the a_m-th canonical vector, that is the vector whose components are all 0 except for the a_m-th entry, which is 1. Then, for all $i = 1, \dots, m-1$, we recursively let $\hat{u}_{j_i}^t := x_{j_i}^t[a_i] \cdot \hat{u}_{j_{i+1}}^t$, $\boldsymbol{\ell}_{j_i}^t := \hat{u}_{j_i}^t \boldsymbol{e}_{a_i}$. Finally, for all $i = 1, \dots, m$, the gradient vector $\boldsymbol{\ell}_{j_i}^t$ is revealed as feedback to the local full-feedback regret minimizer $\mathcal{R}_{j_i}$ at decision point j_i.

Average Policy

When regret minimizers are used to solve a convex-concave saddle point problem (such as a Nash equilibrium in a two-player zero-sum game), only the profile of *average* policies produced by the regret minimizers are guaranteed to converge to the saddle point. For this reason, it is crucial to be able to represent the *average* policy of an agent. Since we are assuming that the structure of the decision problem is only partially known, this operation requires more care in our setting. As we now show, it is possible to modify the algorithm so that the average policy can be extracted.

In order to maintain the average policy, we maintain an additional vector $\bar{\boldsymbol{x}}_j$ at each discovered decision node j. Intuitively, these vector will be populated with entries from the cumulative sum of all partial sequence-form strategies recommended so far by the $\mathcal{R}_j$'s. As soon as j is discovered for the first time (say, time t), we create its $\bar{\boldsymbol{x}}_j$. If j's parent sequence is the empty sequence (that is, j is one of the possible first decision nodes in the SDM process, i.e., j is preceded only by observation nodes), we simply set $\bar{x}_j[a]^t := t/|A_j|$ for all $a \in A_j$. Otherwise, let $p_j = (j', a')$ be the parent sequence of j, and we set $\bar{x}_j[a]^t := \bar{x}_{j'}[a']/|A_j|$ for all $a \in A_j$. Then, at all times t, after the feedback has been received but before the regret update phase has started, we introduce a new *average policy update phase*. In it, we iterate through all the decision nodes j_i that have been discovered so far (that is, the union of all decision points discovered up to time t), in the order they have been discovered. For each of them, we update $\bar{\boldsymbol{x}}_{j_i}$ according to the following rule. Let $\boldsymbol{x}_{j_i}$ be the policy strategy returned by the local full-feedback regret minimizer $\mathcal{R}_{j_i}$. If j_i's parent sequence is the empty sequence, we simply set $\bar{\boldsymbol{x}}_{j_i}^{t+1} := \bar{\boldsymbol{x}}_{j_i}^t + \boldsymbol{x}_{j_i}$ and $r_{j_i}^t := \boldsymbol{x}_{j_i}$. Otherwise, let $p_{j_i} = (j', a')$ be the parent sequence of j, and we set $\bar{\boldsymbol{x}}_{j_i}^{t+1} := \bar{\boldsymbol{x}}_{j_i}^t + r_{j'}^t[a'] \cdot \boldsymbol{x}_{j_i}^t$, and $r_{j_i}^t[a] := r_{j'}^t[a'] \cdot x_{j_i}^t[a]$ for all $a \in A_{j_i}$.

In order to play the average policy, it is enough to play actions proportionally to $\bar{\boldsymbol{x}}_j^{t+1}$ at all discovered decision nodes j, and actions picked uniformly at random at undiscovered decision nodes.

Observation 1. *In all phases, the agent performs an amount of operations at most linear in the number of actions $|A_j|$ at each decision point j discovered up to time t. So, the average policy update phase requires work at most linear in the size of the underlying SDM.*

5 Guarantees on Regret

We now present regret bounds that hold at all times T for our algorithm from Section 4. The bounds do not depend on which rollout variant is chosen at each time t, and different variants can be chosen freely at different times.

Theorem 1 provides a general bound that holds for any choice of local regret minimizers $\mathcal{R}_j$ ($j \in \mathcal{J}$), (non-increasing) stepsizes β^t, and explorations functions h^t at all times t. It will then be the basis for Theorem 2, which provides a way to set the algorithm parameters to achieve sublinear $O(T^{3/4})$ regret.

Theorem 1. *Let $R_j^T(\hat{\boldsymbol{\pi}}_j)$ denote the regret cumulated by the local full-feedback regret minimizer $\mathcal{R}_j$ compared to a generic strategy $\hat{\boldsymbol{\pi}}_j$, and Δ be the maximum range of payoffs that can be selected by the environment at all times. Then, for all $T \geq 1$, $p \in (0,1)$, and $\hat{\boldsymbol{\pi}} \in Q$, with probability at least $1-p$ the regret cumulated by our algorithm satisfies*

$$R^T(\hat{\boldsymbol{\pi}}) \leq \max_{\boldsymbol{q} \in \Pi} \left\{ \sum_{j \in \mathcal{J}} q[p_j] \cdot \max_{\hat{\boldsymbol{\pi}}_j \in \Delta^{|A_j|}} R_j^T(\hat{\boldsymbol{\pi}}_j) \right\} + \frac{\Delta}{\beta^T}(1+\nu)\sqrt{2T \log \frac{2}{p}} + \Delta \sum_{t=1}^{T} \beta^t,$$

where

$$\nu := \sqrt{\frac{1}{T} \sum_{t=1}^{T} \max_{z \in Z} \left\{ \prod_{(j,a) \rightsquigarrow z} \left(\frac{\sum_{a' \in A_j} h^t(j, a')}{h^t(j, a)} \right)^2 \right\}}. \quad (2)$$

(Proofs are in the appendix.) When h^t measures the number of terminal states reachable under any sequence (j, a), the constant ν satisfies $\nu \leq |\Sigma| - 1$. For the uniform exploration function ($h^t(j, a) = 1$ for all j, a), ν is upper bounded by the product of the number of actions at all decision nodes, a polynomial quantity in "bushy" decision processes. In Theorem 2 we operationalize Theorem 1 by showing sensible choices of stepsizes β^t and local regret minimizers $\mathcal{R}_j$.

Theorem 2. *Let the local full-feedback regret minimizers $\mathcal{R}_j$ guarantee $O(T^{1/2})$ regret in the worst case[6], and let $p \in (0,1)$. Furthermore, let the exploration probabilities be $\beta^t := \min\{1, k \cdot t^{-1/4}\}$ for all t, where $k > 0$ is an arbitrary constant. Then, there exists a (decision-problem-dependent) constant c independent of p and T such that for all $T \geq 1$*

$$\mathbb{P}\left[\max_{\hat{\boldsymbol{\pi}} \in \Pi} R^T(\hat{\boldsymbol{\pi}}) \leq c \cdot T^{3/4} \Delta \sqrt{\log \frac{2}{p}} \right] \geq 1 - p.$$

[6] Valid choices include the following algorithms: regret matching (Hart and Mas-Colell 2000), regret matching$^+$ (Tammelin et al. 2015), follow-the-regularized-leader, online mirror descent, exponential weights, hedge, and others.

When h^t is an exact measure of the number of terminal states, c is polynomial in $|\Sigma|$. Otherwise, it is linear in the constant ν defined in (2)*, which depends on the specific exploration functions used.*

Since Theorem 2 guarantees sublinear regret with high probability, our algorithm can be used for all purposes described in Section 3, including computing an approximate Nash equilibrium in a two-player zero-sum extensive-form game whose structure is *a priori* unknown.

6 Empirical Evaluation

In our experiments, we used our algorithm to compute an approximate Nash equilibrium. We compared our method to established model-free algorithms from the multiagent reinforcement learning and computational game theory literature for this setting: *neural fictitious self-play (NFSP)* (Heinrich and Silver 2016), the *policy gradient (PG)* approach of Srinivasan et al. (2018), and the *online variant of Monte-Carlo CFR (online MCCFR)* mentioned in (Lanctot et al. 2009). In line with prior empirical evaluations of those methods, we compare the algorithms on two standard benchmark games: Kuhn poker (Kuhn 1950) and Leduc poker (Southey et al. 2005). The games are reviewed in Appendix C.

We used the implementations of PG and NFSP provided in *OpenSpiel* (Lanctot et al. 2019).They internally use Tensorflow. For PG, we tested the RPG and QPG policy gradient formulations, but not the RM formulation (it performed worst in the original paper (Srinivasan et al. 2018)). We implemented online MCCFR and our algorithm in C++ (online MCCFR is not implemented in OpenSpiel). We ran every algorithm with five random seeds. In the figures below, we plot the average exploitability (a standard measure of closeness to equilibrium) of the players averaged across the five seeds. The shaded areas indicate the *maximum* and *minimum* over the five random seeds. For NFSP we used the hyperparameter recommended by the OpenSpiel implementation. For Kuhn poker, we used the settings for PG that were tuned and found to work the best by Srinivasan et al. (2018)—they are publicly available through OpenSpiel. For PG in Leduc poker, we performed a hyperparameter switch and selected for the two PG plot (RPG and QPG formulation) the best combination hyperparameters (full details are in the appendix). For both online MCCFR and our algorithm, we used RM+ (Tammelin 2014) as the local (full-feedback) regret minimizer. For our algorithm, we only show performance for the on-path-flipping variant. The upfront-flipping variant performed significantly worse and data is available in the appendix. We tested $k \in \{0.5, 1, 10, 20\}$ and set h^t to either the uniform exploration function (h^t constant) or the theoretically-optimal exploration function h^t that measures the number of terminal nodes as explained in Section 4. The performance of the two exploration functions was nearly identical, so in Figure 1 we show our algorithm with the uniform exploration function. We chose $k = 10$ since that performed well on both games. The plots for all other hyperparameter combinations for our algorithm are in the appendix. For online MCCFR, the only hyperparameter is ϵ, which controls the ϵ-greediness of the exploration term added before sampling and outputting the strategy at each time t. We tested $\epsilon = 0.6$ (which was found useful for the different, *self-play* MCCFR algorithm (Lanctot et al. 2009)), 0.1, and 0.0 (which corresponds to pure exploitation); Figure 1 shows the setting that performed best.

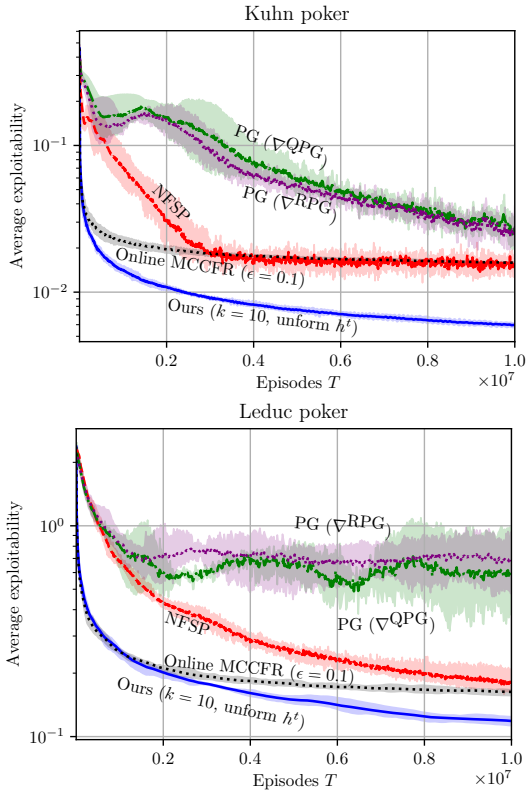

Figure 1: Comparison of the algorithms.

Out of all the algorithms, ours is the only one that guarantees sublinear regret with high probability. This superior guarantee appears to translate into superior practical performance as well. In both benchmark games, our algorithm has lowest exploitability, often by a factor 2x-4x.

7 Conclusions and Future Research

We introduced a new online learning model, which we coined the *interactive bandit model*, that captures sequential decision making. We developed an algorithm that guarantees sublinear $O(T^{3/4})$ regret with high probability in this model, even when the structure of the underlying decision problem is at first unknown to the agent and must be explored as part of the learning process. This is, to our knowledge, the first in-high-probability regret minimizer for this setting. It can be used for multiagent reinforcement learning. Its regret guarantee enables it to be used in any application for which regret minimization is useful: approximating Nash equilibrium or quantal response equilibrium (Ling, Fang, and Kolter 2018; Farina, Kroer, and Sandholm 2019) in two-player zero-sum games, approximating coarse correlated equilibrium in multi-player games (Moulin and Vial 1978; Celli et al. 2020), learning a best response, safe opponent exploitation (Farina, Kroer, and Sandholm 2019), online play against an unknown opponent/environment, etc. It is open whether better than $O(T^{3/4})$ regret is achievable in this important setting.

References

Abernethy, J.; Hazan, E.; and Rakhlin, A. 2008. Competing in the dark: An efficient algorithm for bandit linear optimization. In *In Proceedings of the 21st Annual Conference on Learning Theory (COLT)*.

Abernethy, J. D.; and Rakhlin, A. 2009. Beating the adaptive bandit with high probability. *2009 Information Theory and Applications Workshop* .

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2002. The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal of Computing* 32: 48–77.

Awerbuch, B.; and Kleinberg, R. D. 2004. Adaptive routing with end-to-end feedback: distributed learning and g eometric approaches. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*. ACM.

Azuma, K. 1967. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal* 19(3): 357–367.

Bartlett, P. L.; Dani, V.; Hayes, T.; Kakade, S.; Rakhlin, A.; and Tewari, A. 2008. High-probability regret bounds for bandit online linear optimization. In *Conference on Learning Theory (COLT)*.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up Limit Hold'em Poker is Solved. *Science* 347(6218).

Brown, G. W. 1951. Iterative Solutions of Games by Fictitious Play. In Koopmans, T. C., ed., *Activity Analysis of Production and Allocation*, 374–376. John Wiley & Sons.

Brown, N.; Kroer, C.; and Sandholm, T. 2017. Dynamic Thresholding and Pruning for Regret Minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Brown, N.; and Sandholm, T. 2015. Regret-Based Pruning in Extensive-Form Games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Brown, N.; and Sandholm, T. 2017a. Reduced Space and Faster Convergence in Imperfect-Information Games via Pruning. In *International Conference on Machine Learning (ICML)*.

Brown, N.; and Sandholm, T. 2017b. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

Brown, N.; and Sandholm, T. 2019a. Solving imperfect-information games via discounted regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

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

Bubeck, S.; Lee, Y. T.; and Eldan, R. 2017. Kernel-based methods for bandit convex optimization. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 72–85.

Celli, A.; Marchesi, A.; Bianchi, T.; and Gatti, N. 2020. Learning to Correlate in Multi-Player General-Sum Sequential Games.

Dani, V.; Kakade, S. M.; and Hayes, T. P. 2008. The Price of Bandit Information for Online Optimization. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Even-Dar, E.; Kakade, S. M.; and Mansour, Y. 2009. Online Markov decision processes. *Mathematics of Operations Research* 34(3): 726–736.

Farina, G.; Kroer, C.; and Sandholm, T. 2019. Online Convex Optimization for Sequential Decision Processes and Extensive-Form Games. In *AAAI Conference on Artificial Intelligence*.

Farina, G.; Kroer, C.; and Sandholm, T. 2020. Stochastic Regret Minimization in Extensive-Form Games. In *ArXiV preprint*.

Farina, G.; Schmucker, R.; and Sandholm, T. 2020. Counterfactual-Free Regret Minimization for Sequential Decision Making and Extensive-Form Games. In *Reinforcement Learning in Games (LRG) Workshop at AAAI 2020*.

Flaxman, A.; Kalai, A.; and McMahan, B. 2005. Online Convex Optimization in the Bandit Setting: Gradient Descent Without a Gradient. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

György, A.; Linder, T.; Lugosi, G.; and Ottucsák, G. 2007. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research* 8(Oct): 2369–2403.

Hart, S.; and Mas-Colell, A. 2000. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica* 68: 1127–1150.

Hazan, E. 2016. Introduction to online convex optimization. *Foundations and Trends in Optimization* 2(3-4): 157–325.

Heinrich, J.; Lanctot, M.; and Silver, D. 2015. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, 805–813.

Heinrich, J.; and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121* .

Hoeffding, W. 1963. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association* 58(301): 13–30.

Kash, I. A.; Sullins, M.; and Hofmann, K. 2020. Combining No-regret and Q-learning. In *Autonomous Agents and Multi-Agent Systems*.

Kleinberg, R. 2004. Nearly tight bounds for the continuum-armed bandit problem. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Koller, D.; Megiddo, N.; and von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *Proceedings of the 26th ACM Symposium on Theory of Computing (STOC)*.

Kuhn, H. W. 1950. A Simplified Two-Person Poker. In Kuhn, H. W.; and Tucker, A. W., eds., *Contributions to the Theory of Games*, volume 1 of *Annals of Mathematics Studies, 24*, 97–103. Princeton, New Jersey: Princeton University Press.

Lanctot, M.; Lockhart, E.; Lespiau, J.-B.; Zambaldi, V.; Upadhyay, S.; Pérolat, J.; Srinivasan, S.; Timbers, F.; Tuyls, K.; Omidshafiei, S.; Hennes, D.; Morrill, D.; Muller, P.; Ewalds, T.; Faulkner, R.; Kramár, J.; Vylder, B. D.; Saeta, B.; Bradbury, J.; Ding, D.; Borgeaud, S.; Lai, M.; Schrittwieser, J.; Anthony, T.; Hughes, E.; Danihelka, I.; and Ryan-Davis, J. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games. *CoRR* abs/1908.09453. URL http://arxiv.org/abs/1908.09453.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo Sampling for Regret Minimization in Extensive Games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit Algorithms*. Cambridge University Press. ISBN 9781108486828.

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What game are we playing? End-to-end learning in normal and extensive form games. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Lisỳ, 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.

Matignon, L.; Laurent, G. J.; and Le Fort-Piat, N. 2012. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review* 27(1): 1–31.

Moravčík, 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* .

Moulin, H.; and Vial, J.-P. 1978. Strategically zero-sum games: The class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory* 7(3-4): 201–221.

Neu, G.; Antos, A.; György, A.; and Szepesvári, C. 2010. Online Markov decision processes under bandit feedback. In *Advances in Neural Information Processing Systems*, 1804–1812.

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

Romanovskii, I. 1962. Reduction of a Game with Complete Memory to a Matrix Game. *Soviet Mathematics* 3.

Sandholm, T.; and Crites, R. 1996. Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma. *Biosystems* 37: 147–166. Special issue on the Prisoner's Dilemma. Early version in IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems.

Shalev-Shwartz, S. 2012. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning* 4(2). ISSN 1935-8237. doi:10.1561/2200000018.

Southey, F.; Bowling, M.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' Bluff: Opponent Modelling in Poker. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Srinivasan, S.; Lanctot, M.; Zambaldi, V.; Pérolat, J.; Tuyls, K.; Munos, R.; and Bowling, M. 2018. Actor-Critic Policy Optimization in Partially Observable Multiagent Environments.

Strens, M. 2000. A Bayesian framework for reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Tammelin, O. 2014. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042* .

Tammelin, O.; Burch, N.; Johanson, M.; and Bowling, M. 2015. Solving Heads-up Limit Texas Hold'em. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*.

von Stengel, B. 1996. Efficient Computation of Behavior Strategies. *Games and Economic Behavior* 14(2): 220–246.

Zhou, Y.; Li, J.; and Zhu, J. 2020. Posterior sampling for multi-agent reinforcement learning: solving extensive games with imperfect information. In *Conference on Learning Representations*.

Zimin, A.; and Neu, G. 2013. Online learning in episodic Markovian decision processes by relative entropy policy search. In *Advances in neural information processing systems*, 1583–1591.

Zinkevich, M. 2003. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *International Conference on Machine Learning (ICML)*, 928–936. Washington, DC, USA.

Zinkevich, M.; Bowling, M.; Johanson, M.; and Piccione, C. 2007. Regret Minimization in Games with Incomplete Information. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

A Summary of Online Learning Models

Online learning model	**Assumptions on gradient vector $\boldsymbol{\ell}^t$** (other than bounded norm)	**Revealed feedback**
Full-feedback[2]	unconstrained: $\boldsymbol{\ell}^t \in \mathbb{R}^{\lvert\Sigma\rvert}$	$\boldsymbol{\ell}^t$
Bandit linear optimization	unconstrained: $\boldsymbol{\ell}^t \in \mathbb{R}^{\lvert\Sigma\rvert}$	$(\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t \in \mathbb{R}$
Interactive bandit (this paper)	$u^t : Z \to \mathbb{R}$ choice of payoff at each $z \in Z$ $s_k^t \in S_k$ choice of signal at each $k \in \mathcal{K}$ $\boldsymbol{\ell}^t$ as in Equation (3)	• $z^t \in Z$ terminal state, • $u^t(z^t) = (\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t \in \mathbb{R}$ payoff at z^t

Table 1: Comparison between different online learning models and their assumptions on the gradient vector and revealed feedback.

Observation 2. *In online learning, the environment picks a secret loss function (gradient vector) $\boldsymbol{\ell}^t$ at every iteration t. The gradient vector controls the utility of the player, which is defined as the inner product $(\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t$, where $\boldsymbol{\pi}^t$ is the strategy output by the agent at iteration t. In the interactive bandit model, it is perhaps more natural to think of the environment as picking a choice of signal s_k^t at each observation node $k \in \mathcal{K}$, as well as a choice of payoff $u^t(z)$ for each terminal node $z \in Z$. The utility of the agent is then computed as $u^t(z^t)$, where z^t is the (unique) terminal node that is reached when the agent plays according to $\boldsymbol{\pi}^t \in \Pi$, and the environment sends the signals s_k^t.*

The two points of view can be reconciled. In particular, at every iteration t it is possible to identify a gradient vector $\boldsymbol{\ell}^t$ for the interactive bandit setting such that $u^t(z^t) = (\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t$, as follows. As mentioned in the body, let $\sigma(z)$ be the last sequence (decision node-action pair) on the path from the root of the decision process to terminal node $z \in Z$, and consider the gradient vector $\boldsymbol{\ell}^t$, defined as the (unique) vector such that

$$(\boldsymbol{\ell}^t)^\top \boldsymbol{x} = \sum_{z \in Z} u^t(z) \left(\prod_{(k,s) \rightsquigarrow z} \mathbb{1}[s_k^t = s] \right) x[\sigma(z)] \qquad \forall \boldsymbol{x} \in Q. \tag{3}$$

For each terminal node $z \in Z$ in the argument of the sum, the product in parenthesis is 1 if and only if signals s_k^t picked by the environment are a superset of those on the path from the root to z. Furthermore, when $\boldsymbol{x} = \boldsymbol{\pi}^t$, $x[\sigma(z^t)] = 1$ if and only if the decision maker has decided to play all actions on the path from the root to z^t. So, the only terminal node for which the right hand side of Equation (3) *is nonzero is $z = z^t$. Consequently, $(\boldsymbol{\ell}^t)^\top \boldsymbol{\pi}^t = u^t(z^t)$.*

B Analysis and Proofs

In order to make the proof approachable, at first we will assume that the structure of the decision process is fully known. We will then present and analyze an algorithm that enjoys the properties described in Theorem 1. Finally, we will show that the algorithm can be implemented as described in Section 4, and therefore, that it does not actually need to know the structure of the decision process.

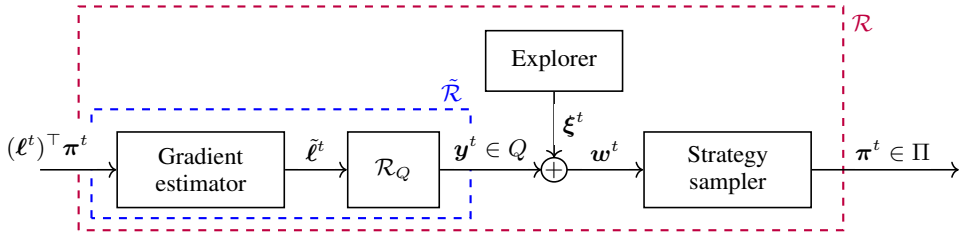

Figure 2: Conceptual construction of our algorithm for the interactive bandit online learning setting.

At a high level, the construction of our interactive bandit regret minimizer works as follows. At each iteration t, we use a full-feedback regret minimizer $\mathcal{R}_Q$ to output a recommendation for the next sequence-form strategy $\boldsymbol{y}^t \in Q$ to be played. Then, we introduce a bias on $\boldsymbol{y}^t$, which can be thought of as an

exploration term. The resulting biased strategy is $\boldsymbol{w}^t \in Q$. Finally, we sample a deterministic policy $\boldsymbol{\pi}^t \in \Pi$ starting from $\boldsymbol{w}^t$. After the playthrough is over and a terminal node $z^t \in Z$ has been reached, we use the feedback (that is, the terminal node z^t reached in the playthrough, together with its utility $u^t(z^t)$), to construct an unbiased estimator $\tilde{\boldsymbol{\ell}}^t$ of the underlying gradient vector that was chosen by the environment.

The algorithm that we have just described accumulates regret from the following three sources.

1. First, it cumulates regret by the full-feedback regret minimizer $\mathcal{R}_Q$. This scales with the norm of the gradient estimators $\tilde{\boldsymbol{\ell}}^t$ as $O(\sqrt{T} \cdot \max_{t=1}^{T} \|\tilde{\boldsymbol{\ell}}^t\|)$.
2. Second, there is a degradation term due to the fact that we do not exactly follow the recommendations from $\mathcal{R}_Q$, and instead bias the recommendations. The biasing is necessary to keep the norm of the gradient estimators under control.
3. Third, there is a regret degradation term due to the fact that we sample a pure strategy starting from the biased recommendation $\boldsymbol{w}^t$.

Our analysis is split into two main conceptual steps. First, we will quantify the regret degradation that is incurred in passing from $\tilde{\mathcal{R}}$ to $\mathcal{R}$ (Figure 2). Specifically, we will study how the regret cumulated by $\tilde{\mathcal{R}}$,

$$\tilde{R}^T(\boldsymbol{\pi}) = \sum_{t=1}^{T} (\boldsymbol{\ell}^t)^\top (\boldsymbol{\pi} - \boldsymbol{y}^t)$$

is related to the regret cumulated by our overall algorithm,

$$R^T(\boldsymbol{\pi}) = \sum_{t=1}^{T} (\boldsymbol{\ell}^t)^\top (\boldsymbol{\pi} - \boldsymbol{\pi}^t).$$

In this case, the degradation term comes from the fact that the strategy output by $\mathcal{R}$, that is, $\boldsymbol{\pi}^t \in \Pi$, is not the same as the one, $\boldsymbol{y}^t$, that was recommended by $\tilde{\mathcal{R}}$, because an exploration term was added.

The second step in the analysis will be the quantification of the regret degradation that is incurred in passing from $\mathcal{R}_Q$ to $\tilde{\mathcal{R}}$ (Figure 2). Specifically, we will study how the regret cumulated by $\tilde{\mathcal{R}}$ is related to the regret

$$\mathcal{R}_Q^T(\boldsymbol{\pi}) = \sum_{t=1}^{T} (\tilde{\boldsymbol{\ell}}^t)^\top (\boldsymbol{\pi} - \boldsymbol{y}^t),$$

which is guaranteed to be $O(T^{1/2})$ with high probability. By summing the degradation bounds, the asymptotic regret guarantee will change from $T^{1/2}$ to $T^{3/4}$, while still retaining a high-probability bound.

Below, we give details about each of the components of our algorithm.

Gradient Estimator The gradient estimator that we use in our algorithm is given in Lemma 1. It is a form of importance-sampling estimator that can be constructed starting from the feedback received by the regret minimizer (that is, the terminal leaf z^t and its corresponding payoff $u^t(z^t)$). It is a particular instantiation of the *outcome sampling* estimator that appeared in the works by Lanctot et al. (2009). We follow the formalization of Farina, Kroer, and Sandholm (2020).

Lemma 1. *Let z^t be the terminal state in which the process ends at iteration t, that is the terminal state reached when the decision maker plays according to the pure strategy $\boldsymbol{\pi}^t$ and the environment plays according to the (partially hidden) pure strategy $\boldsymbol{y}^t$. Also, let $u^t(z^t)$ be the reward of the decision maker upon termination and let $\sigma(z^t)$ denote the last sequence (decision node-action pair) on the path from the root of the decision process to z. Then,*

$$\tilde{\boldsymbol{\ell}}^t := \frac{u^t(z^t)}{w^t[\sigma(z^t)]} \boldsymbol{e}_{z^t}$$

is an unbiased estimator of $\boldsymbol{\ell}^t$, that is, $\mathbb{E}_t[\tilde{\boldsymbol{\ell}}^t] = \boldsymbol{\ell}^t$.

Proof. We will prove the lemma by showing that $\mathbb{E}_t[(\tilde{\boldsymbol{\ell}}^t)^\top \boldsymbol{x}] = (\boldsymbol{\ell}^t)^\top \boldsymbol{x}$. By definition, z^t is the (unique) state such that

$$\pi^t[\sigma(z)] = \prod_{(k,s) \rightsquigarrow z} \mathbb{1}[s_k^t = s] = 1.$$

For all other terminal states z, $\pi^t[\sigma(z)] \cdot \prod_{(k,s)\rightsquigarrow z} \mathbb{1}[s_k^t = s] = 0$. Hence,

$$\begin{aligned}
\mathbb{E}_t\left[(\tilde{\boldsymbol{\ell}}^t)^\top \boldsymbol{x}\right] &= \mathbb{E}_t\left[\frac{u^t(z^t)}{w^t[\sigma(z^t)]}(\boldsymbol{e}_z^\top \boldsymbol{x})\right] \\
&= \mathbb{E}_t\left[\sum_{z\in Z}\left(\pi^t[\sigma(z)] \cdot \prod_{(k,s)\rightsquigarrow z} \mathbb{1}[s_k^t = s]\right)\frac{u^t(z)}{w^t[\sigma(z)]}(\boldsymbol{e}_z^\top \boldsymbol{x})\right] \\
&= \sum_{z\in Z}\frac{\mathbb{E}_t\left[\pi^t[\sigma(z)]\right]}{w^t[\sigma(z)]}u^t(z)\left(\prod_{(k,s)\rightsquigarrow z} \mathbb{1}[s_k^t = s]\right)x[\sigma(z)],
\end{aligned}$$

Using the hypothesis that $\boldsymbol{\pi}^t$ is a (conditionally) unbiased estimator of $\boldsymbol{w}^t$ we further obtain

$$\mathbb{E}_t\left[\pi^t[\sigma(z)]\right] = \mathbb{E}_t[\boldsymbol{\pi}^t][\sigma(z)] = w^t[\sigma(z)]$$

and therefore

$$\mathbb{E}_t\left[(\tilde{\boldsymbol{\ell}}^t)^\top \boldsymbol{x}\right] = \sum_{z\in Z} u^t(z)\left(\prod_{(k,s)\rightsquigarrow z} \mathbb{1}[s_k^t = s]\right)x[\sigma(z)] = (\boldsymbol{\ell}^t)^\top \boldsymbol{x},$$

where we used the definition of $\boldsymbol{\ell}^t$ from (1) in the last equality. □

The Full-Feedback Regret Minimizer Our full-feedback regret minimizer $\mathcal{R}_Q$ is the counterfactual regret minimization (CFR) regret minimizer (Zinkevich et al. 2007). CFR decomposes the task of computing a strategy $\boldsymbol{y}^t \in Q$ for the whole decision process into smaller subproblems at each decision point. The local strategies are then computed via $|\mathcal{J}|$ independent full-feedback regret minimizers $\mathcal{R}_j$ ($j \in \mathcal{J}$), one per decision node, that at each t observe a specific feedback—the *counterfactual gradient*—that guarantees low global regret across the decision process. CFR guarantees $O(T^{1/2})$ regret in the worst case against any strategy $\hat{\boldsymbol{y}}^t \in Q$.

We now review how the counterfactual gradients that are given as feedback at all local regret minimizers $\mathcal{R}_j$ are constructed. Let $\boldsymbol{x}_j^t$, $j \in \mathcal{J}$, be the local strategies output by the $\mathcal{R}_j$'s, and let $\tilde{\boldsymbol{\ell}}^t$ be the gradient vector observed at time t by $\mathcal{R}_Q$. We start by constructing the counterfactual values V_j, indexed over $j \in \mathcal{J}$, which are recursively defined as

$$V_j^t = \sum_{a\in A_j} x_j^t[a]\left(\tilde{\ell}^t[j,a] + \sum_{j'\in\mathcal{J}:p_{j'}=(j,a)} V_{j'}^t\right).$$

Then, for all $j \in \mathcal{J}$ we construct the counterfactual gradient ℓ_j^t by setting $\ell_j^t[a] = \tilde{\ell}^t[j,a] + \sum_{j'\in\mathcal{J}:p_{j'}=(j,a)} V_{j'}^t$ for all $a \in A_j$.

Let $R_j^T(\hat{\boldsymbol{\pi}}_j)$ denote the regret cumulated up to time T by the local full-feedback regret minimizer $\mathcal{R}_j$ compared to strategy $\hat{\boldsymbol{\pi}}_j^t \in \Delta^{|A_j|}$. Then, the regret cumulated by the CFR algorithm is known to satisfy the regret bound in Lemma 2.

Lemma 2 ((Farina, Kroer, and Sandholm 2019))**.** *At all T, for all $\hat{\boldsymbol{\pi}} \in Q$,*

$$R_Q(\hat{\boldsymbol{\pi}}) \le \max_{\boldsymbol{q}\in Q}\left\{\sum_{j\in\mathcal{J}} q[p_j] \cdot \max_{\hat{\boldsymbol{\pi}}_j\in\Delta^{|A_j|}} R_j^T(\hat{\boldsymbol{\pi}}_j)\right\}.$$

Exploration Term The role of the exploration term is to reduce the norm of the gradient estimators. We bias the strategies $\boldsymbol{y}^t$ output by $\mathcal{R}_Q$ by taking a convex combination with the *exploration strategy* $\boldsymbol{\xi}^t \in Q$ defined as the sequence-form strategy that picks action $a \in A_j$ at decision node $j \in \mathcal{J}$ with probability proportional to

$$\xi^t[j,a] \propto \frac{h^t(j,a)}{\sum_{a\in A_j} h^t(j,a)},$$

where h^t is a generic exploration function $\Sigma \to \mathbb{R}_{>0}$.

Let $\sigma(z)$ denote the last sequence (decision node-action pair) encountered on the path from the root of the decision process to terminal node $z \in Z$. A key quantity about $\boldsymbol{\xi}^t$, that will be important to quantify the effect of the biasing on the norm of the gradient estimator, is the following:

$$\rho^t := \max_{z \in Z} \frac{1}{\xi^t[\sigma(z)]} = \max_{z \in Z} \prod_{(j,a) \rightsquigarrow z} \left(\frac{\sum_{a' \in A_j} h^t(j,a')}{h^t(j,a)} \right).$$

At all times t, we construct the biased strategy $\boldsymbol{w}^t$ starting from the unbiased strategy $\boldsymbol{y}^t$ and the exploration strategy $\boldsymbol{\xi}^t$ as

$$\boldsymbol{w}^t := (1 - \beta^t) \cdot \boldsymbol{y}^t + \beta^t \cdot \boldsymbol{\xi}^t. \tag{4}$$

We call the exploration function h^t that assigns the constant value 1 to all sequences the *uniform exploration strategy*. We also study an exploration strategy for which we can prove better worst-case convergence speed. Specifically, we call the exploration function h^t that assigns, to each sequence $(j,a) \in \Sigma$, the number of terminal nodes under (j,a) the *balanced exploration strategy*. For this strategy, the minimum reach probability across all terminal nodes, which is important for our worst-case convergence guarantee, scales linearly in the game size.

Lemma 3 ((Farina, Kroer, and Sandholm 2020))**.** *When h^t is a perfect measure of the number of terminal nodes under each sequence $(j,a) \in \Sigma$, then $\rho^t \leq |\Sigma| - 1$.*

Strategy Sampler The role of the strategy sampler is to provide an *unbiased* estimator $\boldsymbol{\pi}^t \in \Pi$ of $\boldsymbol{w}^t$. In fact, any unbiased estimator can be used in our framework. We start by describing a sampling scheme for sequence-form strategies that is folklore in the literature. In the natural sampling scheme, given a sequence-form strategy $\boldsymbol{q} \in Q$, at all decision nodes $j \in \mathcal{J}$ an action is sampled according to the distribution $q[j,a]/q[p_j] : a \in A_j$.

- *Upfront-flipping strategy sampling* Since by construction $\boldsymbol{w}^t = (1 - \beta^t) \cdot \boldsymbol{y}^t + \beta^t \cdot \boldsymbol{\xi}^t$, sampling from $\boldsymbol{w}^t$ can be done as follows. First, we flip a biased coin, where the probability of heads is $1 - \beta^t$ and the probability of tails is β^t. If heads comes up, we sample actions according to the natural sampling scheme applied to the sequence-form strategy $\boldsymbol{y}^t$ output by $\mathcal{R}_Q$. Otherwise, we sample actions according to the natural sampling scheme applied to the exploration strategy $\boldsymbol{\xi}^t$. We call the strategy sampler just described the *upfront-flipping* sampling scheme.
- *On-path-flipping strategy sampling* We now describe a different sampling scheme, which we coin the *on-path-flipping* sampling scheme. First, we compute the strategy $\boldsymbol{w}^t$ by explicitly taking the convex combination between $\boldsymbol{y}^t$ and the exploration term $\boldsymbol{\xi}^t$. Then we apply the natural sampling scheme for sequence-form strategies to $\boldsymbol{w}^t$. In the on-path-flipping sampling scheme, the exploration and the exploitation are interleaved at each decision node in a counterintuitive way.

Because both sampling schemes are unbiased, our theory, without changes, applies to both of these sampling schemes. Later in this appendix, we report experiments on both of these sampling schemes.

Relationship Between Exploration and Norm of Counterfactual Gradients

The gradient $\tilde{\boldsymbol{\ell}}^t$ that is given to $\mathcal{R}_Q$ as feedback is given in Lemma 1. It has zero entries everywhere, except for the sequence $\sigma(z^t)$. Consequently, the counterfactual values V_j^t are 0 everywhere, except for the decision nodes that were traversed on the path from the root to the terminal node $z^t \in Z$. In turn, this means that all counterfactual gradients are 0, except for the sequences that are traversed. for them, the construction of the counterfactual gradients reduces to the *regret update phase* described in Section 4.

The norms of the counterfactual gradients are maximum at the leaves. In particular, the following is known.

Lemma 4 ((Lanctot et al. 2009; Farina, Kroer, and Sandholm 2020))**.** *Let Δ be the maximum range of payoffs that can be selected by the environment at all times. All counterfactual gradients have norm upper bounded as*

$$\|\boldsymbol{\ell}_j^t\|_2 \leq \frac{\Delta \rho^t}{\beta^t} \quad \forall j \in \mathcal{J}.$$

Furthermore, the loss estimate $\tilde{\boldsymbol{\ell}}^t$ satisfies

$$(\tilde{\boldsymbol{\ell}}^t)^\top (\boldsymbol{x} - \boldsymbol{x}') \leq \frac{\Delta \rho^t}{\beta^t} \quad \forall \boldsymbol{x}, \boldsymbol{x}' \in Q.$$

Relationship Between $\mathcal{R}$ and $\tilde{\mathcal{R}}$

As a first step in our analysis, we establish an important relationship between the regret cumulated by $\mathcal{R}$ and $\tilde{\mathcal{R}}$. Fundamentally, it quantifies the degradation in the regret incurred from playing $\boldsymbol{\pi}^t$ instead of $\boldsymbol{y}^t$ as recommended. The degradation is kept under control by the fact that the expectation of the output $\boldsymbol{\pi}^t$ is close to $\boldsymbol{y}^t$ when β is small. Before we state the central result of this subsection (Proposition 1), we need the following simple bound.

Lemma 5. *Let Δ be the payoff range of the interaction, that is the maximum absolute value of any payoff that can be selected by the environment at any time. At all T, it holds that*

$$\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t-\boldsymbol{w}^t)\leq\Delta\sum_{t=1}^{T}\beta^t.$$

Proof. Using simple algebraic manipulations,

$$\begin{aligned}\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t-\boldsymbol{w}^t)&=\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top\big(\boldsymbol{y}^t-\big((1-\beta^t)\boldsymbol{y}^t+\beta^t\boldsymbol{b}^t\big)\big)\\&=\sum_{t=1}^{T}\beta^t(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t-\boldsymbol{b}^t)\leq\Delta\sum_{t=1}^{T}\beta^t,\end{aligned}$$

where the last inequality follows from the definition of Δ. □

With Lemma 5 we are ready to analyze the relationship between $\mathcal{R}$ and $\tilde{\mathcal{R}}$.

Proposition 1. *At all T, for all $p\in(0,1)$ and $\boldsymbol{\pi}\in\Pi$, with probability at least $1-p$,*

$$R^T(\boldsymbol{\pi})\leq\tilde{R}^T(\boldsymbol{\pi})+\Delta\left(\sqrt{2T\log\frac{1}{p}}+\sum_{t=1}^{T}\beta^t\right).$$

Proof. Introduce the discrete-time stochastic process

$$d^t:=(\boldsymbol{\ell}^t)^\top(\boldsymbol{\pi}^t-\boldsymbol{w}^t),\quad t\in\{1,2,\dots\}.$$

From the online learning model hypotheses, we have that $\boldsymbol{\ell}^t$ is conditionally independent from $\boldsymbol{\pi}^t$ and $\boldsymbol{w}^t$ given all past choices of the algorithm up to time $t-1$, and since $\mathbb{E}_t[\boldsymbol{\pi}^t]=\boldsymbol{w}^t$ by construction, d^t is a martingale difference sequence. Furthermore, some elementary algebra reveals that

$$\begin{aligned}\sum_{t=1}^{T}d^t&=\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{\pi}^t-\boldsymbol{w}^t)\\&=\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{\pi}-\boldsymbol{y}^t)-\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{\pi}-\boldsymbol{\pi}^t)+\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t-\boldsymbol{w}^t)\\&=\tilde{R}^T(\boldsymbol{\pi})-R^T(\boldsymbol{\pi})+\sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t-\boldsymbol{w}^t)\\&\leq\tilde{R}^T(\boldsymbol{\pi})-R^T(\boldsymbol{\pi})+\Delta\sum_{t=1}^{T}\beta^t,\end{aligned}\tag{5}$$

where the last inequality follows from Lemma 5. Since $|\delta^t|\leq\Delta$ for all t, using the Azuma-Hoeffding inequality (Hoeffding 1963; Azuma 1967) we have that

$$\begin{aligned}1-p&\leq\mathbb{P}\left[\sum_{t=1}^{T}d^t\geq-\Delta\sqrt{2T\log\frac{1}{p}}\right]\\&\leq\mathbb{P}\left[\tilde{R}^T(\boldsymbol{\pi})-R^T(\boldsymbol{\pi})+\Delta\sum_{t=1}^{T}\beta^t\geq-\Delta\sqrt{2T\log\frac{1}{p}}\right]\\&=\mathbb{P}\left[R^T(\boldsymbol{\pi})\leq\tilde{R}^T(\boldsymbol{\pi})+\Delta\left(\sqrt{2T\log\frac{1}{p}}+\sum_{t=1}^{T}\beta^t\right)\right],\end{aligned}$$

where the second inequality used (5). □

Relationship Between $\tilde{\mathcal{R}}$ and $\mathcal{R}_Q$

The next proposition establishes the important relationship between the regret cumulated by $\tilde{\mathcal{R}}$ and $\mathcal{R}_Q$. Unlike Proposition 1, where a regret degradation was suffered for playing a strategy different that the recommended one, in this case the regret degradation comes from the fact that the gradient that is observed by $\mathcal{R}_Q$ is different from that observed by $\tilde{\mathcal{R}}$. However, as we will show the degradation is kept under control by the fact that $\tilde{\boldsymbol{\ell}}^t$ is an unbiased estimator of $\boldsymbol{\ell}^t$ by hypothesis.

Proposition 2. *At all T, for all $p \in (0,1)$ and $\boldsymbol{\pi} \in \Pi$, with probability at least $1-p$,*

$$\tilde{R}^T(\boldsymbol{\pi}) \leq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{1}{p}},$$

where

$$\nu := \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\rho^t)^2} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\max_{z\in Z}\left\{\prod_{(j,a)\rightsquigarrow z}\left(\frac{\sum_{a'\in A_j}h^t(j,a')}{h^t(j,a)}\right)^2\right\}}.$$

Proof. Introduce the discrete-time stochastic process

$$d^t := (\boldsymbol{\ell}^t - \tilde{\boldsymbol{\ell}}^t)^\top(\boldsymbol{\pi} - \boldsymbol{y}^t), \quad t \in \{1,2,\dots\}.$$

From the online learning model hypotheses, we have that $\boldsymbol{\ell}^t$ and $\tilde{\boldsymbol{\ell}}^t$ are conditionally independent from $\boldsymbol{y}^t$, and since $\mathbb{E}_t[\tilde{\boldsymbol{\ell}}^t] = \boldsymbol{\ell}^t$ by construction, d^t is a martingale difference sequence. At each t, the conditional range of d^t is upper bounded by

$$|d^t| \leq |(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t - \boldsymbol{\pi})| + |(\tilde{\boldsymbol{\ell}}^t)^\top(\boldsymbol{y}^t - \boldsymbol{\pi})| \leq \Delta + \frac{\Delta}{\beta^t}\rho^t \leq \frac{2\Delta}{\beta^t}\rho^t,$$

where the last inequality follows since $\beta^t \leq 1$ and $\rho^t \geq 1$. Hence, the sum of quadratic ranges for the martingale difference sequence is

$$r := \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\frac{2\Delta}{\beta^t}\rho^t\right)^2} \leq \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\frac{2\Delta}{\beta^T}\rho^t\right)^2} = \frac{2\Delta}{\beta^T}\nu,$$

where we used the fact that the β^t are (weakly) decreasing. Using the Azuma-Hoeffding inequality, we obtain

$$\begin{aligned} 1-p &\leq \mathbb{P}\left[\sum_{t=1}^{T} d^t \leq r\sqrt{\frac{T}{2}\log\frac{1}{p}}\right] \\ &= \mathbb{P}\left[\sum_{t=1}^{T} d^t \leq \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{1}{p}}\right]. \end{aligned} \tag{6}$$

Finally,

$$\begin{aligned} \sum_{t=1}^{T} d^t &= \sum_{t=1}^{T}(\boldsymbol{\ell}^t - \tilde{\boldsymbol{\ell}}^t)^\top(\boldsymbol{y}^t - \boldsymbol{\pi}) \\ &= \sum_{t=1}^{T}(\boldsymbol{\ell}^t)^\top(\boldsymbol{y}^t - \boldsymbol{\pi}) - \sum_{t=1}^{T}(\tilde{\boldsymbol{\ell}}^t)^\top(\boldsymbol{y}^t - \boldsymbol{\pi}) \\ &= \tilde{R}^T(\boldsymbol{\pi}) - R_Q^T(\boldsymbol{\pi}), \end{aligned} \tag{7}$$

and substituting (7) into (6) we have

$$\mathbb{P}\left[\tilde{R}^T(\boldsymbol{\pi}) - R_Q^T(\boldsymbol{\pi}) \leq \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{1}{p}}\right] \geq 1-p.$$

Rearranging the terms inside of the square brackets yields the statement. □

Regret Analysis for the Overall Algorithm

Combining Proposition 2 and Proposition 1 using the union bound lemma, we obtain the following (note that the fractions $1/p$ inside of the logarithms have become $2/p$ as a consequence of the union bound lemma).

Corollary 1. *At all T, for all $p \in (0,1)$ and $\boldsymbol{\pi} \in \Pi$, with probability at least $1-p$,*

$$R^T(\boldsymbol{\pi}) \leq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}(1+\nu)\sqrt{2T\log\frac{2}{p}} + \Delta\sum_{t=1}^{T}\beta^t.$$

where ν is as in Proposition 2.

Proof. From Proposition 2 and Proposition 1, respectively, we have that

$$\frac{p}{2} \geq \mathbb{P}\left[R^T(\boldsymbol{\pi}) \geq \tilde{R}^T(\boldsymbol{\pi}) + \Delta\left(\sqrt{2T\log\frac{2}{p}} + \sum_{t=1}^{T}\beta^t\right)\right]$$
$$\frac{p}{2} \geq \mathbb{P}\left[\tilde{R}^T(\boldsymbol{\pi}) \geq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{2}{p}}\right].$$

Summing the two inequalities and using the union bound, we obtain

$$\begin{aligned}
p &\geq \mathbb{P}\left[R^T(\boldsymbol{\pi}) \geq \tilde{R}^T(\boldsymbol{\pi}) + \Delta\left(\sqrt{2T\log\frac{2}{p}} + \sum_{t=1}^{T}\beta^t\right)\right] + \mathbb{P}\left[\tilde{R}^T(\boldsymbol{\pi}) \leq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{2}{p}}\right] \\
&\geq \mathbb{P}\left[\left(R^T(\boldsymbol{\pi}) \geq \tilde{R}^T(\boldsymbol{\pi}) + \Delta\left(\sqrt{2T\log\frac{2}{p}} + \sum_{t=1}^{T}\beta^t\right)\right) \vee \left(\tilde{R}^T(\boldsymbol{\pi}) \leq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{2}{p}}\right)\right] \\
&\geq \mathbb{P}\left[R^T(\boldsymbol{\pi}) + \tilde{R}^T(\boldsymbol{\pi}) \geq \left(\tilde{R}^T(\boldsymbol{\pi}) + \Delta\left(\sqrt{2T\log\frac{2}{p}} + \sum_{t=1}^{T}\beta^t\right)\right) + \left(R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}\nu\sqrt{2T\log\frac{2}{p}}\right)\right] \\
&= \mathbb{P}\left[\mathbb{R}^T(\boldsymbol{\pi}) \geq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}(\beta^T+\nu)\sqrt{2T\log\frac{2}{p}} + \Delta\sum_{t=1}^{T}\beta^t\right] \\
&\geq \mathbb{P}\left[\mathbb{R}^T(\boldsymbol{\pi}) \geq R_Q^T(\boldsymbol{\pi}) + \frac{\Delta}{\beta^T}(1+\nu)\sqrt{2T\log\frac{2}{p}} + \Delta\sum_{t=1}^{T}\beta^t\right].
\end{aligned}$$

Taking complements yields the statement. □

Theorem 2. *Let the local full-feedback regret minimizers $\mathcal{R}_j$ guarantee $O(T^{1/2})$ regret in the worst case[7], and let $p \in (0,1)$. Furthermore, let the exploration probabilities be $\beta^t := \min\{1, k \cdot t^{-1/4}\}$ for all t, where $k > 0$ is an arbitrary constant. Then, there exists a (decision-problem-dependent) constant c independent of p and T such that for all $T \geq 1$*

$$\mathbb{P}\left[\max_{\hat{\boldsymbol{\pi}} \in \Pi} R^T(\hat{\boldsymbol{\pi}}) \leq c \cdot T^{3/4}\Delta\sqrt{\log\frac{2}{p}}\right] \geq 1-p.$$

When h^t is an exact measure of the number of terminal states, c is polynomial in $|\Sigma|$. Otherwise, it is linear in the constant ν defined in (2)*, which depends on the specific exploration functions used.*

Proof. Fix $T \in \{1, 2, \dots\}$. At all times $t \in \{1, \dots, T\}$, the norm that enter regret minimizer $\mathcal{R}_j$ has norm upper bounded by

$$\|\boldsymbol{\ell}_j^t\| \leq \frac{\Delta}{\beta^t}\rho^t \leq \Delta \max_{t=1}^{T}\{\rho^t\}\max\left\{\frac{1}{k}, 1\right\}T^{1/4},$$

where we used the fact that $\beta^T \geq \min\{k,1\}T^{-1/4}$ for all T. Letting $\tilde{M} := \max_{t=1}^{T}\rho^t$ and $\tilde{k} := \max\left\{\frac{1}{k}, 1\right\}$, we obtain $\|\boldsymbol{\ell}_j^t\| \leq \Delta\tilde{M}\tilde{k}T^{1/4}$.

[7] Valid choices include the following algorithms: regret matching (Hart and Mas-Colell 2000), regret matching$^+$ (Tammelin et al. 2015), follow-the-regularized-leader, online mirror descent, exponential weights, hedge, and others.

Let c_j be the constant in the regret guarantee for the $O(T^{1/2})$ regret of $\mathcal{R}_j$. Since regret guarantees are always with respect to some ball of gradients of bounded norm (say, norm 1) the regret cumulated by $\mathcal{R}_j$ is bounded as $\max_{\hat{\boldsymbol{\pi}}_j} R_j^T(\hat{\boldsymbol{\pi}}_j) \leq c_j \|\boldsymbol{\ell}_j^t\| T^{1/2}$. In other words, we need to keep into account the degradation factor due to the fact that the norms of the gradients constructed through Lemma 1 might exceed the bound for which the regret guarantee for $\mathcal{R}_j$ was given. So, in particular,

$$\max_{\hat{\boldsymbol{\pi}}_j} R_j^T(\hat{\boldsymbol{\pi}}_j) \leq c_j \Big(\Delta \tilde{M} \tilde{k} T^{1/4}\Big) T^{1/2} = \Delta \tilde{M} \tilde{k} c_j T^{3/4},$$

and so

$$\max_{\boldsymbol{q} \in Q} \left\{ \sum_{j \in \mathcal{J}} q[p_j] \cdot \max_{\hat{\boldsymbol{\pi}}_j} R_j^T(\hat{\boldsymbol{\pi}}_j) \right\} \leq \sum_{j \in \mathcal{J}} \max \left\{ 0, \max_{\hat{\boldsymbol{\pi}}_j} R_j^T(\hat{\boldsymbol{\pi}}_j) \right\} \leq \Delta \tilde{M} \tilde{k} \left(\sum_{j \in \mathcal{J}} c_j \right) T^{3/4}. \tag{8}$$

Furthermore,

$$\frac{\Delta}{\beta^T}(1+\nu)\sqrt{2T \log \frac{2}{p}} \leq \Delta \tilde{k} T^{1/4}(1+\nu)\sqrt{2T \log \frac{2}{p}} = \Delta(1+\nu)\tilde{k} T^{3/4}\sqrt{\log \frac{2}{p}}. \tag{9}$$

Finally,

$$\Delta \sum_{t=1}^{T} \beta^t = \Delta \sum_{t=1}^{T} \min\{1, k \cdot t^{-1/4}\} \leq \Delta k \sum_{t=1}^{T} t^{-1/4} \leq \Delta k \int_0^T t^{-1/4} dt = \frac{4\Delta k}{3} T^{3/4}. \tag{10}$$

Substituting the bounds (8), (9), and (10) into the general result of Theorem 1 (specifically, (**??**)), we obtain

$$\begin{aligned} R^T(\boldsymbol{\pi}) &\leq \Delta \tilde{M} \tilde{k} \left(\sum_{j \in \mathcal{J}} c_j \right) T^{3/4} + \Delta(1+\nu)\tilde{k} T^{3/4} \sqrt{\log \frac{2}{p}} + \frac{3\Delta k}{4} T^{3/4} \\ &= \Delta \left(\frac{4k}{3} + (1+\nu)\tilde{k}\sqrt{\log \frac{2}{p}} + \tilde{k}\tilde{M} \sum_{j \in \mathcal{J}} c_j \right) T^{3/4} \\ &\leq \Delta \left(\frac{4k}{3 \log 2} + (1+\nu)\tilde{k} + \frac{\tilde{k}}{\log 2} \tilde{M} \sum_{j \in \mathcal{J}} c_j \right) T^{3/4} \sqrt{\log \frac{2}{p}}, \end{aligned}$$

where we used the fact that $\log(2/p) \geq \log 2$ for all $p \in (0,1)$. Setting c to be the quantity in parentheses and noting that when h^t measures the number of terminal states in each subtree $\nu \leq |\Sigma| - 1$ (Lemma 3), we obtain the statement. □

Remark 1. *Theorem 2 shows that our algorithm achieves $O(T^{3/4}\sqrt{\log(1/p)})$ regret with high probability (specifically, probability at least $1-p$). Indeed, using the properties of logarithms, $\sqrt{\log(2/p)} = \sqrt{\log(1/p) + \log 2} \leq 2\sqrt{\log(1/p)} = O(\sqrt{\log(1/p)})$ for all $p \leq 1/2$.*

Handling the Unknown Structure

As already noted in Section B, only the local full-feedback regret minimizers $\mathcal{R}_j$ on the path from the root to z^t observe a nonzero counterfactual gradient, which is computed as in the *regret update phase* described in Section 4. All other local regret minimizers see a zero gradient, and therefore their regret does not increase. So, we can safely avoid updating those regret minimizers that are not on the path from the root to the most recent terminal leaf.

The upfront-flipping sampling scheme and the on-path-flipping sampling scheme can both be implemented so that the actions are sampled incrementally as the interaction with the environment progresses. In other words, there is no need to sample actions at all decision nodes upfront in order to interact with the environment. In the body of this paper, we only described how to do so for the upfront-flipping sampling scheme, which is conceptually the most important, as it showcases well the difference with, for example, epsilon-greedy exploration used by MCCFR.

C Additional Details about Experiments

Description of Games

Here, we review the two standard benchmark games that we use in the experiments.

In Kuhn poker (Kuhn 1950), two players put an ante worth 1 into the pot at the beginning of the game. Then, each player is privately dealt one card from a deck that contains only three cards—specifically, jack,

queen, and king. Then Player 1 decides to either check or bet 1. If Player 1 checks, Player 2 can decide to either check or raise 1. If Player 2 checks, a showdown occurs. If Player 2 raises, Player 1 can fold—and the game ends with Player 2 taking the pot—or call, at which point a showdown occurs. Otherwise, if the first action of Player 1 is to raise, then Player 2 may fold (the game ends with Player 1 taking the pot) or call, at which point a showdown occurs. In a showdown, the player with the higher card wins the pot and the game ends.

Leduc poker (Southey et al. 2005) is played with a deck of 3 unique ranks, each appearing twice in the deck. There are two rounds in the game. In the first round, all players put an ante of 1 in the pot and are privately dealt a single card. A round of betting then starts. Player 1 acts first, and at most two bets are allowed per player. Then, a card is publicly revealed, and another round of betting takes place, with the same dynamics described above. After the two betting round, if one of the players has a pair with the public card, that player wins the pot. Otherwise, the player with the higher card wins the pot. All bets in the first round are worth 2, while all bets in the second round are 4.

Hyperparameter Tuning for the Policy Gradient Approach by Srinivasan et al. (2018)

We acknowledge the help of some of the original authors (Srinivasan et al. 2018) in tuning hyperparameters for their algorithm on the Leduc poker benchmark.

For both the RPG and the QPG formulation, the following combinations of hyperparameters was tested:

- Critic learning rate: $\{0.001, 0.05, 0.01\}$;
- Pi learning rate: $\{0.001, 0.01, 0.05\}$;
- Entropy cost (used to multiply the entropy loss): $\{0.01, 0.1\}$;
- Batch size (for Q and Pi learning): $\{16, 64, 128\}$;
- Num critics (before each Pi update): $\{16, 64, 128\}$.

In Tables 2 and 3 we report the average exploitability (computed across 5 independent runs for each choice of hyperparameters) for the top 5 choices of hyperparameters for the RPG and QPG formulation, respectively. The experimental results in the body show data for the best combination of eitherparameters (first row of each table).

Critic learning rate	Pi learning rate	Entropy cost	Batch size	Num Critics	Avg. exploitability
0.01	0.05	0.01	128	64	0.448680
0.01	0.05	0.01	64	64	0.451409
0.001	0.01	0.01	16	128	0.472729
0.01	0.01	0.01	128	16	0.493694
0.05	0.01	0.01	128	64	0.494840

Table 2: Performance of PG with RPG gradient formulation using the top 5 combinations of hyperparameters.

Critic learning rate	Pi learning rate	Entropy cost	Batch size	Num Critics	Avg. exploitability
0.01	0.05	0.01	64	128	0.421745
0.01	0.01	0.01	64	16	0.451535
0.001	0.01	0.01	16	128	0.457430
0.01	0.01	0.01	64	128	0.471100
0.05	0.01	0.01	128	64	0.524005

Table 3: Performance of PG with QPG gradient formulation using the top 5 combinations of hyperparameters.

Additional Experiments

In Figures 3 to 6 we study how the uniform and balanced exploration functions compare in Kuhn and Leduc poker, for both the on-path-flipping and the upfront-flipping sampling scheme. We test different values of the exploration multiplier k (Theorem 2), specifically $k \in \{0.5, 1, 10, 20\}$.

The experiments show that the on-path-flipping sampling schemes leads to significantly less variance than the upfront-flipping sampling scheme. Sometimes the latter leads to better average convergence very early on in the learning, but overall has worse convergence in practice.

The experiments show no meaningful different between the uniform and balanced exploration strategies.

Figure 7 shows how MCCFR performed with different exploration parameters.

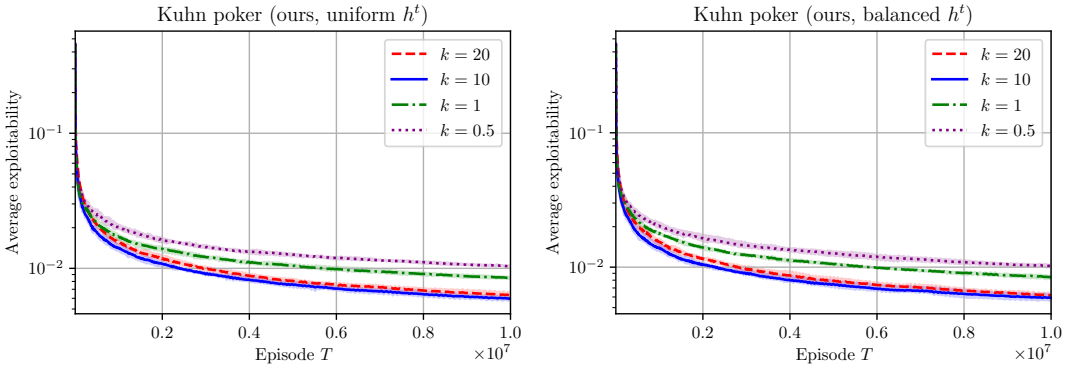

Figure 3: Comparison between uniform and balanced exploration strategies in Kuhn poker, for different values of the exploration multipler k, when using the on-path-flipping sampling scheme.

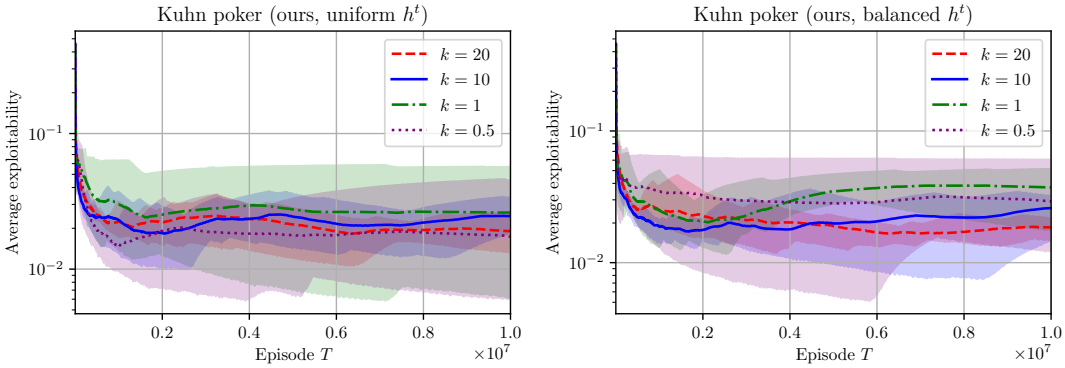

Figure 4: Comparison between uniform and balanced exploration strategies in Kuhn poker, for different values of the exploration multipler k, when using the upfront-flipping sampling scheme.

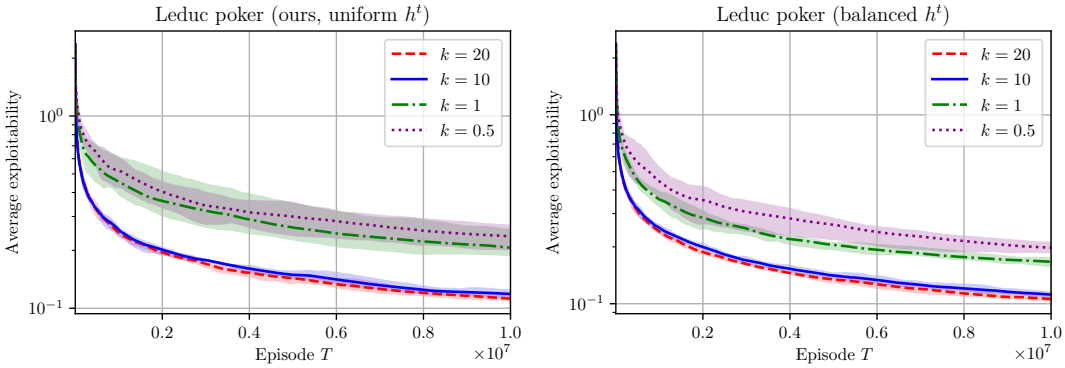

Figure 5: Comparison between uniform and balanced exploration strategies in Leduc poker, for different values of the exploration multipler k, when using the on-path-flipping sampling scheme.

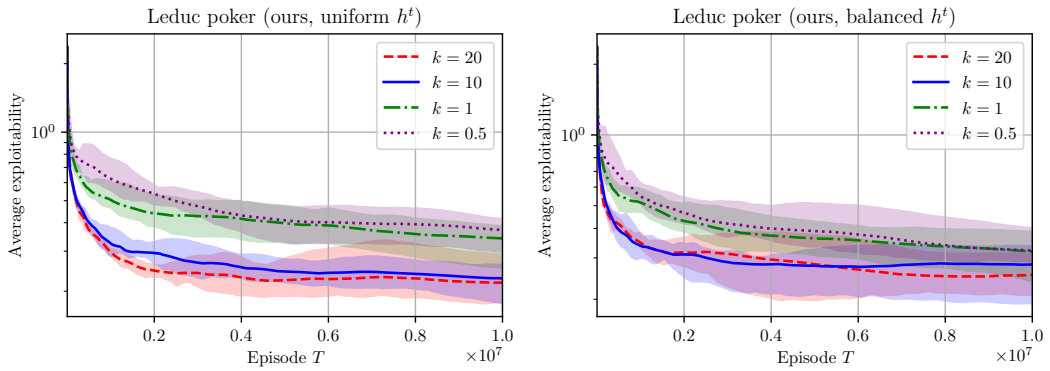

Figure 6: Comparison between uniform and balanced exploration strategies in Leduc poker, for different values of the exploration multipler k, when using the upfront-flipping sampling scheme.

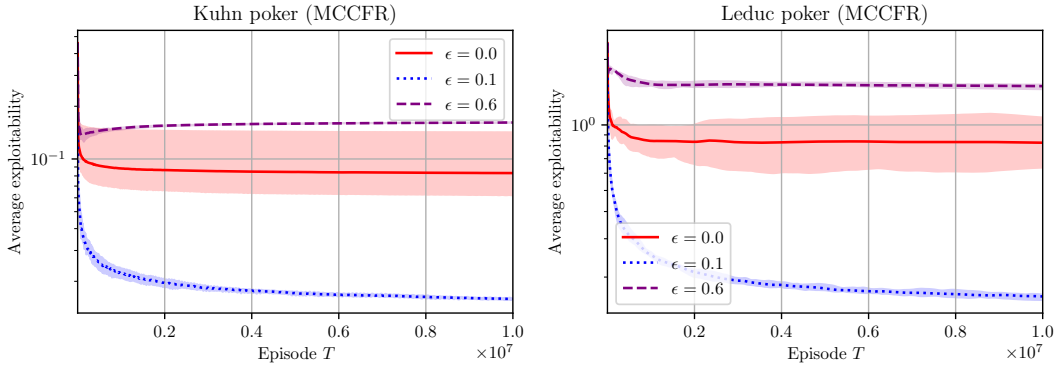

Figure 7: Performance of MCCFR for different amounts of ϵ-greedy exploration.