Easy as ABCs: Unifying Boltzmann Q-Learning and Counterfactual Regret Minimization

Anonymous submission

Abstract

We propose *ABCs* (**A**daptive **B**ranching through **C**hild **s**tationarity), a best-of-both-worlds algorithm combining *Boltzmann Q-learning* (BQL), a classic reinforcement learning algorithm for single-agent domains, and *counterfactual regret minimization* (CFR), a central algorithm for learning in multi-agent domains. ABCs adaptively chooses what fraction of the environment to explore each iteration by measuring the stationarity of the environment's reward and transition dynamics. In Markov decision processes, ABCs converges to the optimal policy with at most an $O(A)$ factor slowdown compared to BQL, where A is the number of actions in the environment. In two-player zero-sum games, ABCs is guaranteed to converge to a Nash equilibrium (assuming access to a perfect oracle for detecting stationarity), while BQL has no such guarantees. Empirically, ABCs demonstrates strong performance when benchmarked across environments drawn from the OpenSpiel game library and OpenAI Gym and exceeds all prior methods in environments featuring partial stationarity.

1 Introduction

The ultimate dream of reinforcement learning (RL) is a general algorithm that can learn in any environment. Nevertheless, present-day RL often requires assuming that the environment is stationary (i.e. that the transition dynamics and rewards do not change over time). When this assumption is violated, many methods, such as Boltzmann Q-learning, fail to learn good policies or even converge at all, both in theory and in practice (Zinkevich et al. 2008; Laurent and Laëtitia Matignon 2011; Brown et al. 2020).

Meanwhile, breakthroughs in no-regret learning, such as counterfactual regret minimization (CFR) (Zinkevich et al. 2008), have led to tremendous progress in imperfect-information, multi-agent games such as Poker and Diplomacy (Moravčík et al. 2017; Brown and Sandholm 2018, 2019b; Meta Fundamental AI Research Diplomacy Team (FAIR) et al. 2022; Bakhtin et al. 2023). Such algorithms are able to guarantee convergence to Nash equilibria in two-player zero-sum games, which are typically not MDPs. However, CFR has poor scaling properties due to the need to perform updates across the entire game tree at *every* learning iteration, as opposed to only across a single trajectory like BQL. As a result, CFR algorithms are typically substantially less efficient than their RL counterparts when used on stationary environments such as MDPs. While Monte Carlo based methods such as *MCCFR* have been proposed to help alleviate this issue (Lanctot et al. 2009), CFR algorithms remain impractical in even toy reinforcement learning environments, such as Cartpole or the Arcade Learning Environment (Sutton and Barto 2018; Bellemare et al. 2013).

We propose ABCs (**A**daptive **B**ranching through **C**hild **s**tationarity), a best-of-both-worlds approach that combines the strengths of both Boltzmann Q-Learning (BQL) and CFR to create a single, versatile algorithm capable of learning in both stationary and nonstationary environments. The key insight behind ABCs is to dynamically adapt the algorithm's learning strategy by quantifying the stationarity of the environment. If an information state (a set of observationally equivalent states) is deemed to be approximately stationary, ABCs performs a relatively cheap BQL-like update. On the other hand, if it is nonstationary, ABCs applies a more expensive CFR-like update to preserve convergence guarantees. This selective updating mechanism enables ABCs to exploit the efficiency of BQL in stationary settings while maintaining the robustness of CFR in nonstationary environments, only exploring the game tree more heavily when necessary.

The precise notion of stationarity that ABCs tests for is *child stationarity*, a weaker notion than Markovian stationarity. In an MDP, the reward and transition functions must remain stationary for the full environment, regardless of the policy chosen by the agent. Instead, child stationarity isolates the transition associated with a specific infostate s and action a and requires only that this transition remain stationary with respect to the policies encountered over the course of learning. We prove that a full CFR update at information states and actions where child stationarity is satisfied is unnecessary to guarantee convergence to equilibria (or to the optimal policy in a single-agent environment), and that a cheaper BQL update suffices, even if the environment as a whole is nonstationary. In MDPs, ABCs asymptotically converges to the optimal policy within a factor of $O(A)$ of the number of updates of BQL, where A is the maximum number of available actions at an infostate. In two-player zero-sum games, ABCs converges to a Nash equilibrium, even while BQL lacks such guarantees, if given access to a perfect oracle for detecting child stationarity. We benchmark ABCs on Cartpole, Leduc poker and Kuhn poker, and show that ABCs has performance comparable to BQL on stationary environments and to CFR on nonstationary ones. Moreover, in environments with both stationary and nonstationary elements, we show that ABCs can outperform both methods.

Related Work

Discovering general algorithms capable of learning in broad environments is a common theme in RL. For example, DQN (Mnih et al. 2015) and its successors (Wang et al. 2016; Bellemare, Dabney, and Munos 2017; Hessel et al. 2018) are capable of obtaining human level performance on a test suite of games drawn from the Arcade Learning Environment, and algorithms such as AlphaZero (Silver et al. 2017) are capable of achieving superhuman performance on perfect information games such as Go, Shogi, and Chess. However, such algorithms still fail on multi-agent imperfect information settings such as Poker (Brown et al. 2020). More recently, several algorithms such as Player of Games (PoG) (Schmid et al. 2021), REBEL (Brown et al. 2020), and MMD (Sokota et al. 2023) have been able to simultaneously achieve reasonable performance on both perfect and imperfect information games. However, unlike ABCs, none of these algorithms adaptively branch information states based on stationarity measurements. As such, they can neither guarantee performance comparable to their reinforcement learning counterparts in MDPs nor efficiently allocate resources in environments which are partially stationary, as ABCs is able to do.

2 Preliminaries

Markov Decision Processes

Classical single-agent RL assumes that the learning environment is a *Markov Decision Process* (MDP). An MDP is defined via a tuple (S, A, T, R, γ). S and A are the set of states and actions, respectively. The state transition function, $T(s' \mid s, a)$, defines the probability of transitioning to state s' after taking action a at state s. $R(r \mid s, a, s')$ is the reward function, which specifies the reward for taking action a at state s, given that we transition to state s'. Importantly, in an MDP, both the transition and the reward function are assumed to be *stationary*. $\gamma \in [0, 1]$ is the discount factor determining the importance of future rewards relative to immediate rewards. In an *episodic MDP*, the agent interacts with the environment across each of separate *episodes* via a *policy*, which is a function $\pi : S \to \Delta A$, mapping states to probabilities over valid actions. In a *Partially Observable Markov Decision Process* (POMDP), the tuple contains an additional element $\mathcal{H}$. In a POMDP, the agent only observes *information states* S while the true hidden states $\mathcal{H}$ of the world remain unknown. The transition and reward dynamics of the environment follow an MDP over the hidden states $\mathcal{H}$, but the agent's policy $\pi : S \to \Delta A$ must depend on the observed infostates rather than hidden states. Nevertheless, there are standard methods for reducing POMDPs into MDPs (see discussion of infostates, below) (Shani, Pineau, and Kaplow 2013).

Finite Extensive-Form Games

While MDPs and POMDPs model a large class of single-agent environments, the stationarity assumption can easily fail in multi-agent environments. Specifically, from a single agent's perspective, the transition and reward dynamics are no longer fixed as they now depend on the other agents who are also learning. To capture these multi-agent environments, we cast our environment as a *finite, potentially imperfect information, extensive-form game with perfect recall*.

In defining this, we have a set of M players (or agents), and an additional *chance player*, denoted by c, whose moves capture any potential stochasticity in the game. There is a finite set $\mathcal{H}$ of possible *histories/hidden states*, corresponding to a finite sequence of states encountered and actions taken by *all* players in the game. Let $h \sqsubseteq h'$ denote that h is a prefix of h', as sequences. The set of *terminal histories*, $Z \subseteq \mathcal{H}$, is the set of histories where the game terminates. For each history $h \in \mathcal{H}$, we let $r_i(h)$ denote the *reward* obtained by player i upon reaching history h.[1] The total *utility* in a given episode for player i after reaching terminal history $z \in Z$ is $u_i(z) = \sum_{h_t \sqsubseteq z} \gamma^t r_i(h_t)$, where h_t denotes the $t + 1$st longest prefix of z (i.e., t indexes a step within an episode). Let $u_i(h, h')$ denote the total utility for player i when reaching history h', starting from history h. Many of the guarantees in this paper focus on two-player zero-sum games, which have the additional property that the sum of the utility functions at any $z \in Z$ for the two players is guaranteed to be 0.

To model imperfect information, we define the set of *infostates* S as a partition of the possible histories $\mathcal{H}$. Let $S(h)$ denote the *information states* (abbreviated infostates) corresponding to history h. There is a *player function* $P : S \to \{1, \ldots, M\} \cup \{c\}$ that maps infostates to the player whose turn it is to move. Each infostate $s \in S$ is composed of states that are observationally equivalent to the player $P(s)$ whose turn it is to play. As is standard, we assume *perfect recall* in that players never forget information obtained through prior states and actions (Zinkevich et al. 2008; Brown and Sandholm 2018; Moravčík et al. 2017; Celli et al. 2020).

Let $\mathcal{A}(s)$ denote the set of possible actions available to player $P(s)$ at infostate s. We denote the policy of player i as $\pi_i : S \to \Delta A$ and let $\pi = \{\pi_1, \cdots \pi_N\}$ denote the *joint policy*. Let $(\pi'_i, \pi_{-i}) = \{\pi_1, \cdots \pi'_i, \cdots \pi_n\}$ denote the policy where all players except player i play according to π and player i plays π'_i. Let $\mathcal{H}_\pi(s)$ denote the distribution of true hidden states corresponding to infostate s, assuming all players play according to joint policy π. We use $T_\pi(h' \mid h, a)$ to denote the probability that h' is the next hidden state after player i plays action a under true hidden state h. We can similarly define the transition probability to infostate s' from hidden state h as $T_\pi(s' \mid h, a) = \sum_{h' \in \mathcal{H}_\pi(s')} \mathbb{1}(S(h') = s') T_\pi(h' \mid h, a)$, and the transition probabilities from infostate to infostate as $T_\pi(s' \mid s, a) = \mathbb{E}_{h \sim \mathcal{H}_\pi(s)}[T_\pi(s' \mid h, a)]$. Analogously, let

[1] Using the discount factor γ, we could equivalently collapse all intermediate rewards into the terminal reward. However, as ABCs makes use of these intermediate rewards during learning, we opt not to do so here.

$R_\pi(r \mid s, a, s')$ denote the probability of obtaining immediate reward r after taking action a at infostate s under joint policy π after transitioning to state s'. In order for the above transitions to always be well defined, we will assume that all policies are fully mixed, in that $\pi(a \mid s) > 0$ for all valid actions a and all infostates s.

3 Unifying Q-Learning and Counterfactual Regret Minimization

The following notation will be useful for describing both BQL and CFR. Let $\eta^\pi(h)$ be the probability of reaching history h if all players play policy π from the start of the game. Let $\eta^\pi_{-i}(h), \eta^\pi_i(h)$ denote the contribution of all players except player i and the contribution of player i to this probability respectively. Analogously, let $\eta^\pi(s) = \sum_h \mathbb{1}(S(h) = h)\eta^\pi(h)$ be the probability of reaching infostate s and $\eta^\pi(h, h')$ be the probability of reaching history h' starting from history h. Since all policies are fully mixed, these probabilities are each guaranteed to be strictly positive. Let Z_s denote the set of all (h, z) such that $S(h) = s$ and $z \in Z$ is a possible terminal history when starting from h.

Boltzmann Q-Learning

Boltzmann Q-learning (BQL) is a variant of the standard Q-learning algorithm, where Boltzmann exploration is used as the exploration policy. Formally, we define the value of a state V and the Q value of an action at that state as:

$$V_i^\pi(s) = \sum_{(h,z)\in Z_s} \frac{\eta^\pi(h)}{\eta^\pi(s)} \eta^\pi(h,z) u_i(h,z)$$

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\substack{s'\sim T_\pi(s'|s,a)\\ r\sim R_\pi(r|s,a,s')}} [r + \gamma V_i^\pi(s')]$$

where the i in the definition of $Q^\pi(s, a)$ refers to the player who plays at state s.

We adapt BQL to the multi-agent setting as follows. At each iteration, BQL first freezes the policies of each agent. It then samples a single trajectory for each agent. These trajectories take the form of (s, a, r, s') tuples, recording the reward r and new infostate s' observed after taking action a at infostate s. Q-values are then updated for each (s, a) pair in the trajectory using a temporal difference (TD) update $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$, where α denotes the learning rate. These average Q-values are stored, and at the end of each iteration n, a new joint policy $\pi^{n+1}(s, a) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'\in A} \exp(Q(s,a')/\tau)}$ is computed by applying Boltzmann exploration on these Q-values, where τ is the temperature parameter. While BQL is known to converge to the optimal policy in MDPs and in perfect information games (given a suitable temperature schedule) (Cesa-Bianchi et al. 2017; Singh et al. 2000), it fails to find the Nash equilibria in imperfect-information multi-agent games such as Poker (Brown et al. 2020).

Counterfactual Regret Minimization

Counterfactual regret minimization (CFR) is a popular method for solving imperfect-information extensive form games and is based on the notion of minimizing *counterfactual regrets*. The counterfactual value for a given infostate is given by $v_i^\pi(s) = \sum_{(h,z)\in Z_s} \eta^\pi_{-i}(h)\eta^\pi(h,z)u_i(h,z)$. The *counterfactual regret* for not taking action a at infostate s is thus given by $r^\pi(s, a) = v_i^{\pi_{(s\to a)}}(s) - v_i^\pi(s)$, where $i = P(s)$ and $\pi_{(s\to a)}$ is identical to policy π except action a is always taken at infostate s. Notice that because $\eta^\pi_i(s) = \eta^\pi_i(h)$ (due to the assumption of perfect recall), the counterfactual values are simply the standard RL value functions normalized by the opponents' contribution to the probability of reaching s; that is, $v_i^\pi(s) = \frac{V_i^\pi(s)}{\eta^\pi_{-i}(s)}$[2]. This was first pointed out by (Srinivasan et al. 2018). As with multi-agent BQL, at each iteration, we freeze all agents' policies before traversing the entire game tree and computing the counterfactual regrets for each infostate, action pair. A new current policy π^{n+1} is then computed using a regret minimizer, typically regret matching (Hart and Mas-Colell 2000) or Hedge (Arora, Dekel, and Tewari 2012) on the cumulative regrets $R^n(s, a) = \sum_n r^{\pi^n}(s, a)$, and the process continues. A learning procedure has *vanishing regret* if $\lim_{N\to\infty} \frac{1}{N} R^N(s, a) = 0$ for all s, a. In two player, zero-sum games, (Zinkevich et al. 2008) prove that CFR has vanishing regret, and by extension, that the average policy converges to a Nash equilibrium of the game. In general games, computing a Nash equilibrium is known to be computationally hard (Daskalakis, Goldberg, and Papadimitriou 2009; Chen, Deng, and Teng 2009), but CFR still guarantees convergence to a slightly weaker solution concept known as the coarse-correlated equilibrium (Morrill et al. 2021).

External Sampling MCCFR with Hedge

External sampling MCCFR (ES-MCCFR) is a popular variant of CFR which uses sampling. While vanilla CFR computes counterfactual regrets for the entire game tree at each iteration, Monte Carlo CFR (MCCFR) approaches rely on sampling to get unbiased estimates of these regrets (Lanctot et al. 2009). For each player i, ES-MCCFR samples a single action for every infostate which players other than i act on, but explores all actions for player i. Let W denote the set of terminal histories reached during this sampling process. For any infostate s, let $h(s)$ denote the corresponding history that was reached during exploration (there can only be one due to perfect recall). For each visited infostate s, the sampled counterfactual regret, $\tilde{r}_i(s, a) = \tilde{v}_i^{\pi_{(s\to a)}}(s, a) - \tilde{v}_i^\pi(s, a) = \sum_{z\in W} u_i(h, z)(\eta^\pi_i(h(s'), z) - \eta^\pi_i(h(s), z))$, is added to the cumulative regret, where s' is the infostate reached after taking action a at infostate s. After this process has been completed for each player, a new joint policy is computed using Hedge on the new cumulative regrets, and the procedure continues. Under Hedge, the joint policy at

[2]Intuitively, the counterfactual regret upper bounds how much an agent could possibly "regret" not playing action a at infostate s if she modified her policy to maximize the probability that state s is encountered. As such, the normalization constant excludes player i's probability contribution to reaching s under π because she is assumed to be playing an alternative policy that is counterfactually trying to reach infostate s.

episode $n+1$ is given by $\pi^{n+1}(s,a) = \frac{e^{R^n(s,a)/\tau_n}}{\sum_{a' \in A(s)} e^{R^n(s,a')/\tau_n}}$, where τ_n is a temperature hyperparameter. As the Hedge regret minimizer is a softmax, it is invariant to any additive constants in the exponent. This means that we can replace the sampled counterfactual regrets with the sampled counterfactual values, i.e., $\tilde{v}_i^\pi(s,a) = \sum_{z \in W} u_i(z) \cdot \eta_i^\pi(h(s'), z)$, and similarly with the cumulative regrets.

BQL vs. CFR

The most important difference between CFR and BQL is in regard to which infostates are updated at each iteration of learning. By default, BQL only performs updates along a single trajectory of encountered infostates ("trajectory based"). In a game of depth D, this means that there will be at most $O(D)$ updates per iteration. By contrast, CFR and most of its variants are typically [3] not trajectory-based learning algorithms because many infostates that are updated are not on the path of play. In particular, CFR (Zinkevich et al. 2008) requires performing counterfactual value updates at every infostate, which can require up to $O(A^D)$ updates per iteration. While MCCFR reduces this by performing Monte-Carlo updates, in practice variants such as ES-MCCFR still require making updates to infostates that grows in number exponentially in the depth of the tree.

Despite the differences between BQL and CFR, the two algorithms are intimately linked. (Brown et al. 2019, Lemma 1) first pointed out that the sampled counterfactual values in ES-MCCFR have a strong connection to the notion of Q-values, in that the sampled counterfactual values $\tilde{v}_i^\pi(s,a)$ have an alternative interpretation as Monte-Carlo based estimates of the Q-value $Q^\pi(s,a)$. Thus, ES-MCCFR with Hedge has a gradient update that ends up being very similar to BQL, with two exceptions. Firstly, BQL estimates its Q-values using bootstrapping, thus computing the *average Q-value* across all previous policies whereas ES-MCCFR uses a Monte-Carlo update and thus updates based on the *current* policy of all players. Secondly, BQL chooses a policy based on a softmax over the *average* Q-values, while CFR(Hedge) uses a softmax over *cumulative* Q-values. As not every action is tested in BQL, the average Q-values may be based on a different number of samples for each action, making it difficult to convert average Q-values into cumulative Q-values.

4 Learning our ABCs

As described in Section 3, while the updates that CFR(Hedge) and BQL perform at each infoset are quite similar, BQL learns over trajectories, whereas CFR expands most (if not all) of the game tree at *every infostate*. This exploration is necessary in nonstationary environments where the Q-values may change over time but wasteful in stationary environments. As a concrete example, consider the following game. Half of the game is comprised of a standard game of poker, and half of the game is a "dummy" game identical to poker except that the payoff is always 0. In the "poker" subtree, BQL will fail to converge because this subtree is not an MDP. However, on the "dummy" subtree of the game, CFR will continue to perform full updates and explore the entire subtree, even though simply sampling trajectories on this "dummy" subtree would be sufficient for convergence.

What if we could formulate an algorithm that could itself discover when it needed to expand a child node, and when it could simply sample a trajectory? The hope is that this algorithm could closely match the performance of RL algorithms in stationary environments while retaining performance guarantees in nonstationary ones. Moreover, one can easily imagine general environments containing elements of both – playing a game of Poker followed by a game of Atari, for example – where such an algorithm could yield better performance than either CFR or BQL.

Child Stationarity

We now define the important concept of *child stationarity*, which is a relaxation of the standard Markov stationarity requirement. Let $\sigma : \Delta\Pi$ denote a distribution over joint policies.

Definition 4.1 (Child stationarity)**.** An infostate s and action a satisfy *child stationarity with respect to distributions over joint policies σ and σ'* if and only if $\mathbb{E}_{\pi \in \sigma}[T_\pi(h' \mid s,a)] = \mathbb{E}_{\pi \in \sigma'}[T_\pi(h' \mid s,a)]$ and $\mathbb{E}_{\pi \sim \sigma}[R_\pi(r \mid s,a,s')] = \mathbb{E}_{\pi \sim \sigma'}[R_\pi(r \mid s,a,s')]$, where s' denotes the infostate observed after playing a at s.

An infostate action pair, (s,a), satisfies child stationarity for policy distributions σ and σ' if the local reward and state transition functions remain stationary despite a change in policy distribution, and even if other portions of the environment, possibly including ancestors of s, are nonstationary. Importantly, the transition function must remain constant with respect to the true, hidden states and not merely the observed infostates.[4] By definition, an MDP satisfies child stationarity for all infostates s and actions a with respect to all possible distributions over joint policies σ, σ'.

The utility of child stationarity can be seen as follows. While MDPs and perfect information games can be decomposed into subgames that are solved independently from each other (Tesauro et al. 1995; Silver et al. 2016; Sutton and Barto 2018), the same is not true for their imperfect-information counterparts (Brown et al. 2020). Finding transitions that satisfy child stationarity allows for natural decompositions of the environment that can be solved independently from the rest of the environment. As such, child stationarity forms a core reason for why ABCs is able to unify BQL and CFR into a single algorithm. Define $G^\sigma(s,a)$ as a game whose initial hidden state is drawn from $\mathbb{E}_{\pi \in \sigma}[T_\pi(h' \mid s,a)]$ and then played identically to G, the original game, from that point onward. $G^\sigma(s,a)$ is analogous to the concept of Public Belief States introduced by ReBeL (Brown et al. 2020, Section 4).

[3]In theory, variants such as OS-MCCFR can learn with merely $O(D)$ updates per iteration, but the variance of such methods is typically too high for practical usage.

[4]We give a more detailed discussion of the similarities and differences between child stationarity and Markovian stationarity in Appendix B.

Algorithm 1 Returns True if s, a is not child stationary.

Precondition: infostate s, action a, child hidden states $h_{1:N}$ (states following (s, a)), total episodes N, significance level $\alpha_s > 0$.

function DETECT($s, a, r_{1:N}, h_{1:N}, \alpha_s$)
  $X_1 \leftarrow \{r_n, h_n \mid n \in 1 \cdots \lfloor N/2 \rfloor\}$
  $X_2 \leftarrow \{r_n, h_n \mid n \in \lfloor N/2 \rfloor + 1 \cdots N\}$
  pval $\leftarrow$ ChiSquared(X_1, X_2)
return $pval < \alpha_s$

Theorem 4.2. *If s, a satisfy child stationarity with respect to σ, σ', then $G^{\sigma}(s, a) = G^{\sigma'}(s, a)$. Furthermore, if π^* is a Nash equilibrium of $G^{\sigma}(s, a)$, then π^* is also a Nash equilibrium of $G^{\sigma'}(s, a)$.*

Proof. By the definition of child stationarity, $\mathbb{E}_{\pi \in \sigma}[T_{\pi}(h' \mid s, a)] = \mathbb{E}_{\pi \in \sigma'}[T_{\pi}(h' \mid s, a)]$. Thus, $G^{\sigma}(s, a)$ and $G^{\sigma}(s, a)$ are informationally equivalent, as all players have the same belief distribution over hidden states at the beginning of the game and will thus have identical equilibria. □

Detecting Child Stationarity

How can we measure child stationarity in practice? Notice that the definition only requires that the transition and reward dynamics stay fixed with respect to two particular distributions of joint policies, not all of them. Let $\pi_1, \pi_2, \cdots \pi_N$ be the policies used by our learning procedure for each iteration of learning from $1 \cdots N$. Define σ_A^N as the uniform distribution over $\{\pi_1 \cdots \pi_{\lfloor N/2 \rfloor}\}$ and σ_B^N as the uniform distribution over $\{\pi_{\lfloor N/2 \rfloor + 1} \cdots \pi_N\}$. At iteration N, Algorithm 1 directly tests child stationarity with respect to distributions over joint policies σ_A^N, σ_B^N as follows: for each infostate s and action a, it keeps track of the rewards and true hidden states that follow playing action a at s. Then, it runs a Chi-Squared Goodness-Of-Fit test (Pearson 1900) to determine whether the distribution of true hidden states has changed between the first and second half of training. We adopt child stationarity as the null hypothesis. While Algorithm 1 is not a perfect detector, Theorem 4.3 shows that in the limit, Algorithm 1 will asymptotically never claim that a given s, a, σ_A, σ_B satisfies child stationarity when it does not, assuming (s, a) is queried infinitely often. However, the Chi Squared test retains a fixed rate of false positives given by the significance level α_s, so Algorithm 2 will incorrectly reject the null hypothesis of child stationarity with some positive probability.

Theorem 4.3. *As $|X_1|, |X_2| \to \infty$, the probability that Algorithm 1 falsely claims s, a satisfies child stationarity when it does not goes to 0.*

Proof. This statement is directly implied by the fact that the Chi-Squared Goodness-of-Fit test guarantees that the Type II error vanishes in the limit (Pearson 1900). □

Algorithm 2 ABCs

Precondition: Q, CNT initialized to 0, discount factor γ, total episodes N. Initial observation and hidden states s_0, h_0. $H = \varnothing$. Significance value p, exploration ϵ

for $n \leftarrow 1$ to N **do**
  ABCs($h_0, s_0, Q, \gamma, n, a_0, H, p, \epsilon$) ▷ a_0 is null

function GETCHILD(h, s, a, π, Q, γ)
  Sample $h', s' \sim T^{\pi}(h' \mid h, a), S(h')$
  Sample $r \sim R^{\pi}(r \mid s, a, s')$
  $a^* \leftarrow \arg\max_a Q(s', a)$
  $\nabla_{s,a} \leftarrow (r + \gamma Q(s', a^*)) - Q(s, a)$
  return $h', s', a^*, \nabla_{s,a}$

function ABCS($h, s, Q, \gamma, n, a_{opt}, H, p, \epsilon$)
  if h is terminal **then return** 0
  $\pi^n \leftarrow \texttt{softmax}(Q(s, *), \tau = 1/(\tau_n \cdot \text{CNT}(s)))$
  CNT(s) $\leftarrow$ CNT(s) + 1
  Sample a_{traj} from $\pi^n(s)$ with ϵ-exploration
  for $a \in A(s)$ **do**
    $h', s', a^*, \nabla_{s,a} \leftarrow$ GetChild($h, s, a, \pi^n, Q, \gamma$)
    Append r, h' to $H(s, a)$
    if DETECT($s, a, H(s, a), p$) OR $a = a_{traj}$ **then**
      $\nabla_{s',a^*} \leftarrow$ ABCs($h', s', Q, \gamma, t, a^*, H, p, \epsilon$)
      if s is not stationary **then**
        $\nabla_{s,a} \leftarrow \nabla_{s,a} + \text{CNT}(s') \cdot \nabla_{s',a^*}$
    $Q(s, a) \leftarrow Q(s, a) + \frac{1}{\text{CNT}(s)} \nabla_{s,a}$
  $a_{opt} \leftarrow \arg\max Q(s, *)$
  return $\nabla_{s,a_{opt}}$

The ABCs Algorithm

We are now ready to learn our ABCs. At a high level, ABCs chooses between a BQL update and a CFR update for each $Q(s, a)$ value based on whether or not it satisfies child stationarity with respect to the joint policies encountered over the course of training. One can view ABCs from two equivalent perspectives. From one perspective, ABCs runs a variant of ES-MCCFR, except that at stationary infostates, where it is unnecessary to run a full CFR update, it defaults to a cheaper BQL update. Alternatively, one can view ABCs as defaulting to a variant of BQL until it realizes that an infostate is too nonstationary for BQL to converge. At that point, it backs off to more expensive CFR style updates. We provide the formal description of ABCs in Algorithm 2. In the multi-agent setting, ABCs freezes all players' policies at the beginning of each learning iteration and then runs independently for each player, exactly as multi-agent BQL and ES-MCCFR does.

Where ABCs Runs Updates As described in Section 3, one major difference between BQL and ES-MCCFR is the number of infostates each algorithm updates per learning iteration. While BQL updates only infostates encountered along a single trajectory, ABCs adaptively adjusts the proportion of the game tree it expands by measuring child stationarity at each (s, a). Concretely, at every infostate encountered (regardless of its stationarity), ABCs chooses an action a_{traj} from its policy (with ϵ-exploration) and recursively runs itself on the resulting successor state s'. For all other actions $a' \neq a$, it tests whether the tuple (s, a') satisfies child stationarity. Unless the null is rejected, it prunes all further updates on that subtree for this iteration. Otherwise, when the detector returns True, it also recursively runs on the successor state sampled from $T(s' \mid s, a')$. As such, if all actions a fail the child stationarity test, then ABCs will expand all child nodes, just like ES-MCCFR, but if the environment is an MDP and all s, a satisfy child stationarity (according to the detector), then ABCs will only expand one child at each state and thus approximate BQL style trajectory updates. With sufficient ϵ-exploration (and because at least one action a_{traj} will be expanded for each infostate s), ABCs satisfies the requirement of Theorem 4.3 that every infostate s will be visited infinitely often in the limit.

Q-value updates In terms of the actual Q-value update that ABCs performs, recall that in Section 3, we describe two differences between BQL and ES-MCCFR, namely that 1) ES-MCCFR updates its Q-values based on the *current* policy (while BQL performs updates based on the *average* policy), and 2) ES-MCCFR chooses its policy as a function of *cumulative* Q-values, whereas BQL uses *average* Q-values. ABCs unifies these updates, as follows. To determine which update to make, ABCs measures the level of child stationarity for a given (s, a) pair, as it does when choosing *which* infostates to update. If (s, a) satisfies child stationarity, then ABCs does a regular BQL update. If not, ABCs adds a nonstationarity correction factor $\text{CNT}(s') \cdot \nabla_{s',a'}$ to the Q-value update, correcting for the difference between the instantaneous Q-values sampled via Monte-Carlo sampling and the average bootstrapped Q-values. This correction exactly recovers the update function of MAX-CFR, a variant of ES-MCCFR described in more detail in Appendix C.

In regard to cumulative vs. average Q-values, we force ABCs to test each action upon reaching an infostate, and thus update $Q(s, a)$ for all a. This allows us to set the temperature to recover a scaled multiple of the cumulative Q-values. Specifically, by setting a temperature of $\frac{1}{\tau_n \cdot CNT(s)}$, where $CNT(s)$ counts the number of visits to s, we recover a softmax over the cumulative Q-values scaled by τ_n, just like with CFR(Hedge). BQL is well known to converge to the optimal policy in an MDP as long as we are greedy in the limit and visit every state, action pair infinitely often, which is guaranteed with an appropriate choice of τ_n and ϵ-schedule (Singh et al. 2000; Cesa-Bianchi et al. 2017).

Convergence Guarantees We formally prove the following theorems describing the performance of an idealized form of ABCs in MDPs and two-player zero-sum games. Theorem 4.4 proves that a minor variant of ABCs finds a Nash equilibrium in a two-player zero-sum game (assuming access to a perfect oracle). Additionally, Theorem 4.5 proves that ABCs converges no slower than an $O(A)$ factor compared to BQL in an MDP without the assumption of a perfect oracle and using only the detector described in Section 4. This is possible because CFR updates will also converge to the optimal policy in MDPs. The full proofs are included in the supplementary material.

Theorem 4.4. *Assume that Algorithm 2 tracks separate $Q^{stat}(s, a)$ and $Q^{nonstat}(s, a)$ values for stationary and nonstationary updates and has access to a perfect oracle for detecting child stationarity. Then, the average policy $\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \pi_n$ in Algorithm 2 converges to a Nash equilibrium in a two-player zero-sum game with high probability.*

Theorem 4.5. *Given an appropriate choice of significance levels p, Algorithm 2 asymptotically converges to the optimal policy with only a worst-case $O(A)$ factor slowdown compared to BQL in an MDP, where A is the maximum number of actions available at any infoset in the game.*

In our practical implementation of ABCs, we make several simplifications compared to the assumptions required in theory. Firstly, as described in Algorithm 2, we only track a single Q-value for both stationary and nonstationary updates. Secondly, we use a constant significance value for all infostates. Finally, we only perform the stationarity check with probability 0.05 at each iteration, significantly reducing the time spent on these checks. In our experiments, we find that these changes simplify the implementation and yield good results in practice. Additionally, since assuming access to a perfect oracle is unrealistic in practice, we use the detector described in Algorithm 1, which guarantees asymptotically vanishing Type II error, but nevertheless retains some probability of a false positive in the limit. Finally, we allow for different temperature schedules for stationary and nonstationary infostates.

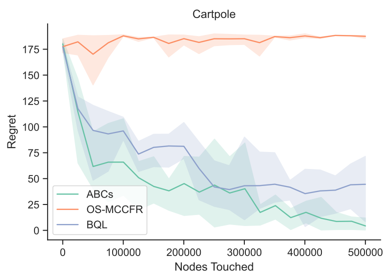
5 Experimental Results

We evaluate ABCs across single and multi-agent settings drawn from both the OpenSpiel game library (Lanctot et al. 2019) and OpenAI Gym (Brockman et al. 2016), benchmarking it against MAX-CFR, BQL (Sutton and Barto 2018), OS-MCCFR (Lanctot et al. 2009), and ES-MCCFR (Lanctot et al. 2009) (when computationally feasible). All experiments with ABCs were run on a single 2020 Macbook Air M1 with 8GB of RAM. We use a single set of hyperparameters and run across three random seeds, with 95% error bars included. Hyperparameters and detailed descriptions of the environments are given in Appendix E. Auxiliary experiments with TicTacToe are also included in Appendix F. All plots describe regret/exploitability, so a lower line indicates superior performance.

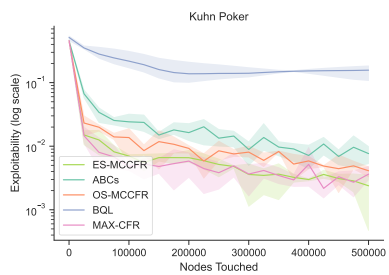
Cartpole We evaluate ABCs on OpenAI Gym's Cartpole environment, a classic benchmark in reinforcement learning[5]. Figure 1a shows that ABCs performs comparably to BQL and substantially better than OS-MCCFR. ES-MCCFR is infeasible to run due to the large state space of Cartpole.

Weighted Rock Paper Scissors Weighted rock-paper-scissors is a classic nonstationary environment in which

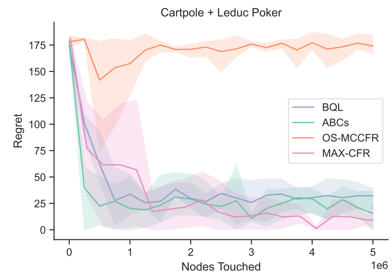
[5]We modify Cartpole slightly to ensure that it satisfies the Markovian property. See Appendix E for details.



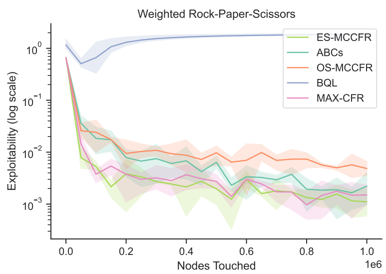
(a) Cartpole



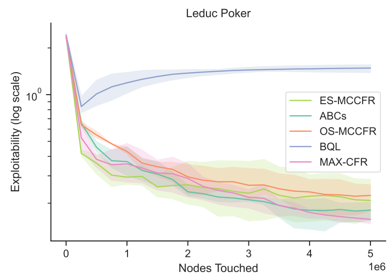
(c) Kuhn Poker



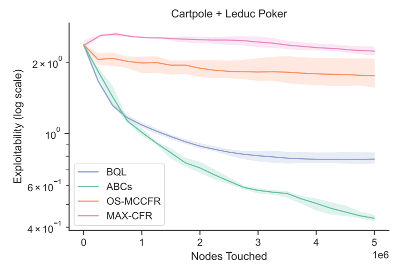
(e) Cartpole (Stacked Environment)



(b) Weighted RPS



(d) Leduc Poker



(f) Leduc (Stacked Environment)

Figure 1: ABCs matches the performance of BQL on stationary environments like Cartpole (a) and the performance of CFR methods on non-stationary environemnts like weighted rock-paper-scissors (b), Kuhn poker (c), and Leduc poker(d). Additionally, on a partially stationary environment, ABCs outperforms both BQL and CFR, being the only algorithm capable of efficiently solving both the Cartpole (e) and Leduc poker (f) portion of the stacked environment.

rock-paper-scissors is played, but there is twice as much reward if a player wins with Rock. Figure 1b demonstrates that while BQL fails to learn the optimal policy, ABCs performs comparably to the standard CFR benchmarks.

Kuhn and Leduc Poker Kuhn and Leduc Poker are simplified versions of the full game of poker which operate as classic benchmarks for equilibrium finding algorithms. As illustrated in Figures 1c and 1d, BQL fails to converge on either Kuhn or Leduc Poker as they are not Markovian environments, and we show similar results for a wide variety of hyperparameters in Appendix F. However, both ABCs and MCCFR converge to a Nash equilibrium of the game, with ABCs closely matching the performance of the standard MCCFR benchmarks.

A Partially Nonstationary Environment We also test ABCs's ability to adapt to a partially nonstationary environment, where its strengths are most properly utilized. Specifically, we consider a stacked environment, where each round consists of a game of Cartpole followed by a game of Leduc poker against another player. Figures 1e and 1f show that BQL does not achieve the maximum score in this combined game, failing to converge on Leduc poker. Although CFR based methods are theoretically capable of learning both games, MAXCFR wastes substantial time conducting unnecessary exploration of the game tree in the larger Cartpole setting, failing to converge on Leduc. OS-MCCFR similarly fails to learn either environment, though in this case, the issue preventing convergence on Leduc relates more to variance induced by importance sampling on such a deep game tree rather than unnecessary exploration. As such, ABCs outperforms all CFR variants and BQL by better allocating resources between the stationary and nonstationary portions of the environment.

6 Conclusion

We have introduced ABCs, a unified algorithm which combines aspects of both Boltzmann Q-Learning and Counterfactual Regret Minimization. We prove that ABCs can simultaneously match the performance of BQL up to an $O(A)$ factor in MDPs while guaranteeing convergence to Nash equilibria in two-player zero-sum games, assuming access to a perfect oracle. To our knowledge, this is the first such algorithm of its kind. Our experiments confirm the efficacy of our child stationary detector and show that ABCs has comparable performance to BQL on stationary environments and CFR on nonstationary ones, and is able to exceed both algorithms in partially nonstationary domains.

We give a number of limitations of ABCs, which we leave to future work. While Theorem 4.4 guarantees that ABCs converges to a Nash equilibrium in two-player zero-sum games, it both requires access to a perfect oracle and does not bound the rate of convergence. Additionally, due to computational restrictions, we benchmark ABCs only on relatively small environments solvable using only tabular methods. For future work, we hope to scale ABCs to larger settings such as the Arcade Learning Environment (Bellemare et al. 2013) and larger games such as Go or Poker using function approximation and methods such as DQN (Mnih et al. 2013) or DeepCFR (Brown et al. 2019).

References

Arora, R.; Dekel, O.; and Tewari, A. 2012. Online Bandit Learning Against an Adaptive Adversary: From Regret to Policy Regret. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 1747–1754. Omnipress. ISBN 978-1-4503-1285-1.

Arora, S.; Hazan, E.; and Kale, S. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1): 121–164.

Bakhtin, A.; Wu, D. J.; Lerer, A.; Gray, J.; Jacob, A. P.; Farina, G.; Miller, A. H.; and Brown, N. 2023. Mastering the Game of No-Press Diplomacy via Human-Regularized Reinforcement Learning and Planning. In *ICLR*.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 449–458. JMLR.org.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47: 253–279.

Billingsley, P. 2012. *Probability and Measure*. John Wiley & Sons.

Brafman, R. I.; and Tennenholtz, M. 2002. R-MAX-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 17057–17069. Curran Associates, Inc.

Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep Counterfactual Regret Minimization. In *International Conference on Machine Learning*, 793–802. PMLR.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for Heads-up No-Limit Poker: Libratus Beats Top Professionals. *Science*, 359(6374): 418–424.

Brown, N.; and Sandholm, T. 2019a. Solving Imperfect-Information Games via Discounted Regret Minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1829–1836.

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

Celli, A.; Marchesi, A.; Farina, G.; and Gatti, N. 2020. No-Regret Learning Dynamics for Extensive-Form Correlated Equilibrium. *Advances in Neural Information Processing Systems*, 33: 7722–7732.

Cesa-Bianchi, N.; Gentile, C.; Lugosi, G.; and Neu, G. 2017. Boltzmann Exploration Done Right. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Chen, X.; Deng, X.; and Teng, S.-H. 2009. Settling the Complexity of Computing Two-Player Nash Equilibria. *Journal of the ACM (JACM)*, 56(3): 1–57.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing*, 39(1): 195–259.

Freund, Y.; and Schapire, R. E. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Journal of Computer and System Sciences*, volume 55, 119–139.

Hart, S.; and Mas-Colell, A. 2000. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica*, 68(5): 1127–1150.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Hu, J.; and Wellman, M. P. 2003. Nash Q-Learning for General-Sum Stochastic Games. *J. Mach. Learn. Res.*, 4(null): 1039–1069.

Lanctot, M.; Lockhart, E.; Lespiau, J.-B.; Zambaldi, V.; Upadhyay, S.; Pérolat, J.; Srinivasan, S.; Timbers, F.; Tuyls, K.; Omidshafiei, S.; et al. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo Sampling for Regret Minimization in Extensive Games. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Laurent, G. J.; and Laëtitia Matignon, N. L. F.-P. 2011. The world of Independent learners is not Markovian. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, 15(1): 55–64. 10.3233/KES-2010-0206. hal-00601941.

Meta Fundamental AI Research Diplomacy Team (FAIR); Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; Jacob, A. P.; Komeili, M.; Konath, K.; Kwon, M.; Lerer, A.; Lewis, M.; Miller, A. H.; Mitts, S.; Renduchintala, A.; Roller, S.; Rowe, D.; Shi, W.; Spisak, J.; Wei, A.; Wu, D.; Zhang, H.; and Zijlstra, M. 2022. Human-Level Play in the Game of Diplomacy by Combining Language Models with Strategic Reasoning. *Science*, 378(6624): 1067–1074.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

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 No-Limit Poker. *Science*, 356(6337): 508–513.

Morrill, D.; D'Orazio, R.; Sarfati, R.; Lanctot, M.; Wright, J. R.; Greenwald, A.; and Bowling, M. 2021. Hindsight and Sequential Rationality of Correlated Play.

OpenAI. 2019. OpenAI Five. *https://openai.com/blog/openai-five/*.

Pearson, K. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302): 157–175.

Schmid, M.; Moravcik, M.; Burch, N.; Kadlec, R.; Davidson, J.; Waugh, K.; Bard, N.; Timbers, F.; Lanctot, M.; Holland, Z.; Davoodi, E.; Christianson, A.; and Bowling, M. 2021. Player of Games. *CoRR*, abs/2112.03178.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Shani, G.; Pineau, J.; and Kaplow, R. 2013. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27: 1–15.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587): 484–489.

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

Singh, S.; Jaakkola, T.; Littman, M. L.; and Szepesvári, C. 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38: 287–308.

Sokota, S.; D'Orazio, R.; Kolter, J. Z.; Loizou, N.; Lanctot, M.; Mitliagkas, I.; Brown, N.; and Kroer, C. 2023. A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two-Player Zero-Sum Games. In *ICLR*.

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. *Advances in Neural Information Processing Systems*, 31.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press, second edition edition. ISBN 978-0-262-03924-6.

Tammelin, O. 2014. Solving Large Imperfect Information Games Using CFR+.

Tesauro, G.; et al. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3): 58–68.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature*, 575(7782): 350–354.

von Neumann, J. 1928. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1): 295–320.

Wang, Z.; Schaul, T.; Hessel, M.; van Hasselt, H.; Lanctot, M.; and de Freitas, N. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1995–2003.

Zhang, H.; Lerer, A.; and Brown, N. 2022. Equilibrium finding in normal-form games via greedy regret minimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 9484–9492. Abstract Note: We extend the classic regret minimization framework for approximating equilibria in normal-form games by greedily weighing iterates based on regrets observed at runtime. Theoretically, our method retains all previous convergence rate guarantees. Empirically, experiments on large randomly generated games and normal-form subgames of the AI benchmark Diplomacy show that greedy weights outperforms previous methods whenever sampling is used, sometimes by several orders of magnitude.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems*, 1729–1736.

A Nash Equilibria and Exploitability

We formally describe a Nash equilibrium and the concept of exploitability. Let $\pi : S \to \Delta(\mathcal{A})$ denote the *joint policy* of all players in the game. We assume that π has full support: for all actions $a \in \mathcal{A}(s)$, $\pi(a \mid s) > 0$. Let Σ denote the space of all possible joint policies, with Σ_i defined accordingly. Let $u_i(\pi)$ denote the expected utility to player i under joint policy π. A joint policy π is a *Nash equilibrium* of a game if for all players i, $\pi_i \in \arg\max_{\pi_i' \in \Sigma_i} u_i(\pi_i', \pi_{-i})$. Similarly, policy π is an *ϵ-Nash equilibrium* if for all players i, $u_i(\pi) \geq \max_{\pi_i' \in \Sigma_i} u_i(\pi_i', \pi_{-i}) - \epsilon$.

In a two-player zero-sum game, playing a Nash equilibrium guarantees you the minimax value; in other words, it maximizes the utility you receive given that your opponent always perfectly responds to your strategy. *Exploitability* is often used to measure the distance of a policy π from a Nash equilibrium, bounding the worst possible losses from playing a given policy. The total exploitability of a policy π is given by $\sum_{i \in M} \max_{\pi_i' \in \Sigma_i} u_i(\pi_i', \pi_{-i}) - u_i(\pi)$. In a two-player zero-sum game, the exploitability of a policy π is given by $\max_{\pi_1' \in \Sigma_1} u_1(\pi_1', \pi_2) + \max_{\pi_2' \in \Sigma_2} u_2(\pi_1, \pi_2')$.

B Markov Stationarity vs. Child Stationarity

In this section, we highlight the relationship between Markov stationarity and child stationarity. An MDP satisfies child stationarity at all infostates and actions s, a and with respect to all possible distributions over joint policies σ, σ' (see Section 4).

Here we highlight an environment that does not satisfy Markov stationarity but satisfies child stationarity. Consider an environment with a single infostate and two possible hidden states. Furthermore, let $h_n = h_{n \mod 2}$. Such an environment is not stationary in the Markov sense, as the hidden states change with each episode. However, defining σ_A, σ_B as the empirical distribution of joint policies across the first and second half of timesteps as described in Section 4, such an environment would (asymptotically) satisfy child stationarity as the distribution of hidden states for the first and second half would approach $\left(\frac{1}{2}, \frac{1}{2}\right)$ in the limit. As shown in Section D, child stationarity, despite being weaker than Markov stationarity still allows BQL to converge to the optimal policy.

C The MAX-CFR Algorithm

MAX-CFR is a variant of ES-MCCFR that ABCs reverts to if the environment fails child stationarity at every possible s, a (i.e., is "maximally nonstationary"). MAX-CFR can also be viewed as a bootstrapped variant of *external-sampling MCCFR* (Lanctot et al. 2009). See Algorithm 3.

Connecting ABCs, MAX-CFR, and ES-MCCFR

To make explicit the connection between MAX-CFR and traditional external-sampling MCCFR (Lanctot et al. 2009) we adopt the Multiplicative Weights (MW) algorithm (also known as Hedge) as the underlying regret minimizer. In MW, a player's policy at iteration $n+1$ is given by

$$\pi^{n+1}(s, a) = \frac{e^{\tau_n R^n(s,a)}}{\sum_{a' \in A(s)} e^{\tau_n R^n(s,a')}},$$

where $R^n(s, a)$ denotes the cumulative regret at iteration n, for action a at infostate s. Hedge also includes a tunable hyperparameter τ_n, which may be adjusted from iteration to iteration. Letting $\tau_n = 1$ across all iterations, we recover the softmax operator on cumulative regrets as a special case of Hedge.

We will assume there are no non-terminal rewards r_t, and use a discount factor of $\gamma = 1$, both of which are standard for the two-player zero-sum environments on which CFR is typically applied. We adopt $\tau_n = 1$ for simplicity, but note that the following equivalences hold for any schedule of τ_n.

We first present Boot-CFR, a "bootstrapped" version of external-sampling MCCFR that is identical to the original ES-MCCFR algorithm. To make the comparison between Boot-CFR and MAX-CFR clear, we highlight the difference between the two algorithms in cyan.

Lemma C.1. *Multiplicative Weights / Hedge is invariant to the choice of cumulative regrets or cumulative rewards/utility.*

Proof. It is well-known and easy to verify that Multiplicative Weights / Hedge is shift-invariant. That is,

$$\text{Hedge}(x) = \text{Hedge}(x + c).$$

Over N iterations, the cumulative counterfactual regret of not taking action a at infostate s is given by

$$R^N(s, a) = \frac{1}{N} \sum_{n=1}^{N} v_n^{\pi^n_{(s \to a)}}(s) - v_n^{\pi^n}(s),$$

Algorithm 3 MAX-CFR

Precondition: Q, CNT initialized to 0, discount factor γ, total episodes N. Initial observation and hidden states s_0, h_0.

```
for n ← 1 to N do
    MAXCFR(h_0, s_0, Q, γ, n)

function MAXCFR(h, s, Q, γ, n)
    if h is terminal then return 0
    π^n(s) ← softmax(Q(s, *), τ = 1/CNT(s))
    CNT(s) ← CNT(s) + 1

    for a ∈ A(s) do
        α ← 1/CNT(s)
        h', s', a', ∇_{s,a} ← GetChild(h, s, a, n, Q, γ)
        ∇_{s',a'} ← f(h', s', Q, γ, n)

        ∇_{s,a} ← ∇_{s,a} + CNT(s') · ∇_{s',a'}
        Q(s, a) ← Q(s, a) + α∇_{s,a}

    a* ← arg max_a Q(s, a)
    return ∇_{s,a*}
```

Algorithm 4 Boot-CFR

Precondition: Same preconditions as MAX-CFR (Algorithm 3).

for $n \leftarrow 1$ to N **do**
  BOOTCFR(h_0, s_0, Q, γ, n, BOOTCFR)

function MODGETCHILD(h, s, a, π, Q, γ)
  Sample $h', s' \sim T^\pi(h' \mid h, a), S(h')$
  Sample $r \sim R^\pi(r \mid s, a, s')$
  return h', s', r

function BOOTCFR(h, s, Q, γ, n)
  if h is terminal **then**
    for $a \in A(s)$ **do**
      $\Delta_{s,a} \leftarrow 0$
    return

  $\pi^n(s) \leftarrow \texttt{softmax}\Big(Q(s,*), \tau = \frac{1}{\text{CNT}(s)}\Big)$
  $\text{CNT}(s) \leftarrow \text{CNT}(s) + 1$

  for $a \in A(s)$ **do**
    $\Delta_{s,a} \leftarrow 0$
    $h', s', r \leftarrow$ ModGetChild($h, s, a, \pi^n, Q, \gamma$)
    for $a' \in A(s')$ **do**
      $\nabla_{s,a,a'} \leftarrow r + \gamma Q(s', a')) - Q(s, a)$

    $Q_{\text{old}} \leftarrow Q(s', *)$
    BOOTCFR(h', s', Q, γ, n)
    $Q_{\text{new}} \leftarrow Q(s', *)$

    $\pi^n_{\text{eval}}(s') \leftarrow \texttt{softmax}\Big(Q_{\text{old}}(s', *), \tau = \frac{1}{\text{CNT}(s')-1}\Big)$
    for $a' \in A(s')$ **do**
      $\Delta_{s,a} \leftarrow \Delta_{s,a} + \pi^n_{\text{eval}}(s', a') \cdot \Big[\frac{1}{\text{CNT}(s)}\left(\nabla_{s,a,a'} + \text{CNT}(s') \cdot \Delta_{s',a'}\right)\Big]$
    $Q(s, a) \leftarrow Q(s, a) + \Delta_{s,a}$
return

where $\pi^n_{s \to a}$ is identical to π^n, except that action a is always taken given infostate s. Since the second term does not depend on the action a, this is equivalent to using Hedge on the cumulative counterfactual rewards

$$\frac{1}{N}\sum_{n=1}^{N} v_n^{\pi^n_{(s\to a)}}(s),$$

by the shift-invariance of Hedge. □

Lemma C.2. *Boot-CFR and external-sampling MCCFR use the same current policy and traverse the game tree in the same way.*

Proof. First, note that game-tree traversal is identical to external-sampling MCCFR by construction. For a given player, we branch all their actions and sample all opponent actions from the current policy. Additionally, while external-sampling MCCFR uses softmax on cumulative regrets, by Lemma C.1, this is equivalent to softmax on cumulative rewards. Note that the policy in Boot-CFR is given by

$$\pi^n(s) = \texttt{softmax}\left(Q(s,*), \tau = \frac{1}{\text{CNT}(s)}\right) = \texttt{softmax}\left(Q(s,*) \cdot \text{CNT}(s)\right).$$

As $Q(s,a)$ is the average reward from taking action a at infostate s, it follows that $Q(s,a) \cdot \text{CNT}(s,a)$ is the cumulative reward. Further, since we branch all actions anytime we visit an infoset, then $\text{CNT}(s,a) = \text{CNT}(s)$ and the policy in Boot-CFR is equivalent to a softmax on cumulative rewards. □

Theorem C.3. *Boot-CFR and external-sampling MCCFR are identical.*

Proof. Boot-CFR and external-sampling MCCFR are identical regarding traversal of the game tree and computation of the current policy (Lemma C.2). Thus, it suffices to show that the updates to cumulative/average rewards are identical.

Let W denote the set of terminal states that are reached on a given iteration of Boot-CFR or external-sampling MCCFR. For any $z \in W$, we define $u_i(z)$ to be the reward for player i at terminal history z. Consider current history h, current infostate s, and action a. Let h' be the history reached on the current iteration after taking action a. For external-sampling MCCFR, the sampled counterfactual regret of taking action a at infostate s is given by

$$\tilde{r}(s,a) = \sum_{z \in W} u_i(z)(\eta_i^\pi(h', z) - \eta_i^\pi(h, z)).$$

By Lemma C.1, when using Hedge, we can consider only the sampled counterfactual rewards

$$r(s,a) = \sum_{z \in W} u_i(z) \cdot \eta^\pi(h', z).$$

While the sampled counterfactual reward is typically only defined for non-terminal observations, for the sake of the proof, we extend the definition to terminal observations, letting $r(s,a) = u_i(z)$, where z is the terminal history corresponding to observation h.

We will prove that $\Delta_{s,a} = \frac{r(s,a) - Q(s,a)}{\text{CNT}(s)}$ for all s that we visit on a given iteration and all actions a.

We proceed by induction. First, we have this equivalence for all terminal nodes, since $\Delta_{s,a} = 0$ at a terminal node. Now, consider an arbitrary infostate s and action a, and suppose that $\Delta_{s',a'} = \frac{r(s',a') - Q(s',a')}{\text{CNT}(s',a')}$ for all actions a' and s' reachable from s after playing action a. By our inductive hypothesis, we have

$$\begin{aligned}
&\nabla_{s,a,a'} + \text{CNT}(s') \cdot \Delta_{s',a'} \\
&= Q(s',a') - Q(s,a) + \text{CNT}(s') \cdot \frac{r(s',a') - Q(s',a')}{\text{CNT}(s')} \\
&= r(s',a') - Q(s,a).
\end{aligned}$$

Additionally, we have

$$\begin{aligned} r(s,a) &= \sum_{z \in W} u_i(z) \cdot \eta_i^{\pi}(h', z) \\ &= \sum_{z \in W} u_i(z) \cdot \left[\sum_{a' \in A(s')} \pi(a'|s') \cdot \eta_i^{\pi}(h''_{a'}, z) \right] \\ &= \sum_{a' \in A(s')} \pi(a'|s') \sum_{z \in W} u_i(z) \eta_i^{\pi}(h''_{a'}, z) \\ &= \sum_{a' \in A(s')} \pi(a'|s') \cdot r(s', a'), \end{aligned}$$

where $h''_{a'}$ is the history reached after playing action a' at history h'. It follows that

$$\begin{aligned} \Delta_{s,a} &= \sum_{a' \in A(s')} \frac{1}{CNT(s)} \cdot \pi(a'|s') \cdot [\nabla_{s,a,a'} + \text{CNT}(s') \cdot \Delta_{s',a'}] \\ &= \sum_{a' \in A(s')} \frac{1}{CNT(s)} \cdot \pi(a'|s') \cdot [r(s',a') - Q(s,a)] \\ &= \frac{1}{\text{CNT}(s)} \cdot \left[-Q(s,a) + \sum_{a' \in A(s')} \pi(a'|s') \cdot r(s',a') \right] \\ &= \frac{r(s,a) - Q(s,a)}{CNT(s)}, \end{aligned}$$

as desired. The Q-update becomes

$$\begin{aligned} Q(s,a) &\leftarrow Q(s,a) + \frac{r(s,a) - Q(s,a)}{CNT(s)} \\ &= \frac{(CNT(s) - 1) \cdot Q(s,a) + r(s,a)}{CNT(s)}, \end{aligned}$$

and corresponds precisely to updating the average Q-value with the external-sampling sampled counterfactual reward. □

Corollary C.4. *MAX-CFR and External-Sampling MCCFR are identical, up to minor differences in the calculation of an action's expected utility/reward.*

Proof. One can verify that there is a single difference between Boot-CFR and MAX-CFR, where the line highlighted in cyan in Algorithm 4 is replaced by

$$\pi^n_{\text{eval}}(s') \leftarrow \texttt{hardmax}(Q_{\text{new}}(s', *) \cdot \text{CNT}(s')),$$

where `hardmax` places all probability mass on the entry/action with the highest value. Thus, instead of calculating the sampled reward of taking action a at infostate s with respect to the current softmax policy, MAX-CFR calculates the sampled reward with respect to a greedy policy that always selects the action with the greatest cumulative reward/smallest cumulative regret. This greedy policy is taken with respect to the new Q-values after they have been updated on the current iteration.

Importantly, the current policy and cumulative policy of all players is still calculated using softmax on cumulative rewards. It is only the calculation of sampled rewards that relies on this new hardmax policy. □

Finally, we can relate MAX-CFR to ABCs. We have the following simple relationship.

Theorem C.5. *Up to ϵ-exploration, ABCs reduces to MAX-CFR in the case where all (s,a) pairs are detected as non-stationary.*

Proof. This follows from the definition of Algorithm 2, where removing the ϵ-exploration recovers MAX-CFR. □

Theorem C.6. *With high probability, MAX-CFR minimizes regrets at a rate of $O\left(\frac{1}{\sqrt{N}}\right)$.*

Proof. Written in terms of Q-value notation, the local regrets that are minimized by CFR(Hedge) are given by

$$r^N(s,a) = \eta^{\pi}_{-i}(s)\big(Q^{\pi}(s,a) - \mathbb{E}_{a \sim \pi(s)}[Q^{\pi}(s,a)]\big).$$

Let

$$r^{N+}(s,a) = \max\big(0, r^N(s,a)\big)$$

Define $\pi^n_{s,a}$ as the policy in which:

1. All players except player $P(s)$ follow π at all states in the game,
2. Player $P(s)$ plays as necessary to reach s and plays action a at s, and
3. Player $P(s)$ plays the action with the highest Q value at s', where s' is the state directly following s.

Policy $\pi^n_{s,a}$ is identical to the counterfactual target policy that CFR(Hedge) follows, except that $\pi^n_{s,a}$ alters the policy at the successor state s' to follow the action with the highest Q value instead of sampling an action according to π^n. Analogously, define the modified MAXCFR local regrets as

$$r^N_{MAXCFR}(s,a) = \eta^{\pi}_{-i}(s)\Big(Q^{\pi^n_{s,a}}(s,a) - \mathbb{E}_{a \sim \pi(s)}[Q^{\pi}(s,a)]\Big)$$
$$r^{N+}_{MAXCFR}(s,a) = \max\big(0, r^N_{MAXCFR}(s,a)\big)$$

These regrets are identical to the standard CFR regrets $r^N(s,a)$, except for the fact that the target policy is modified to $\pi^n_{s,a}$.

Lemma C.7. *For any s', π and any possible Q values,*

$$\max_a Q(s',a) - \mathbb{E}_{a \sim \pi}[Q(s',a)] \geq 0.$$

Proof. This follows immediately from the properties of the max operator. □

Lemma C.8. $r^N(s,a) \leq r^N_{MAXCFR}(s,a)$.

Proof. By definition of $\pi^n_{s,a}$, we can write:

$$\begin{aligned} Q^{\pi^n_{s,a}}(s,a) &= Q^{\pi}(s,a) \\ &\quad + \mathbb{E}_{s' \sim T^{\pi}(s'|s,a)}\Big[\max_{a'} Q(s',a') - \mathbb{E}_{a' \sim \pi}[Q(s',a']\Big]. \end{aligned}$$

Since the second term is strictly nonnegative by Lemma C.7, we have $R^N_{MAXCFR}(s,a) > R^N(s,a)$. □

Lemma C.9. *Let Δ be the difference between the lowest and highest possible rewards achievable for any player in G. Let $N_{s,a}$ denote the number of times MAX-CFR has visited s,a. We have $R^N_{MAXCFR}(s,a) \leq \frac{\Delta}{\sqrt{N_{s,a}}}$.*

Proof. Since we choose all policies over a softmax, all policies have full support and $\eta^{\pi}_{-1}(s) > 0$ for all s. Thus, $N_{s,a} = N$ in MAX-CFR. Noting that MAX-CFR chooses its policy $\pi^n_{s,a}$ proportionally to $\exp\big(r^N_{MAXCFR}(s,a)\big)$, it follows from the Multiplicative Weights algorithm (Freund and Schapire 1997; Arora, Hazan, and Kale 2012), that the regrets given by $r^N_{MAXCFR}(s,a)$ after N iterations are bounded above by $\frac{\Delta}{\sqrt{N}}$ with high probability. □

Notice that like CFR, the guarantee of Lemma C.9 holds *regardless* of the strategy employed at all other s, a in the game, so long as the policy at iteration n chooses action a at state s proportionally to $\exp\big(r^N_{MAXCFR}(s,a)\big)$. Since Lemma C.8 shows that local regrets for MAXCFR form an upper bound on the standard CFR regrets (over the sequence of joint policies $\pi^1 \cdots \pi^N$), we can apply (Zinkevich et al. 2008, Theorem 3) along with Lemma C.9 to show that the overall regret of the game is,

$$r^G \leq \|\mathcal{S}\| \, \|\mathcal{A}\| \max_{s,a} r^{N+}_{MAXCFR}(s,a)$$

$$\leq \|\mathcal{S}\| \, \|\mathcal{A}\| \frac{\Delta}{\sqrt{N}} = O\left(\frac{1}{\sqrt{N}}\right)$$

□

D Convergence of ABCs

Theorem 4.4. *Assume that Algorithm 2 tracks separate $Q(s,a)$ values for stationary and nonstationary updates and has access to a perfect oracle for detecting child stationarity. Then, the average policy $\lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \pi_n$ in Algorithm 2 converges to a Nash equilibrium in a two-player zero-sum game with high probability.*

Proof. Call our current game G. We will prove convergence to a Nash equilibria in G by showing the average policy in Algorithm 2 has vanishing local regret. This is sufficient to show the algorithm as a whole has no-regret and thus converges to a Nash equilibrium (Zinkevich et al. 2008, Theorem 2).

We first define a perfect oracle. At every episode N, a perfect oracle has full access to the specification of game G and the joint policy, $\pi_1 \cdots \pi_N$, of all players in each episode 1 through N. As per Section 4, define σ^N_A as the uniform distribution over $\{\pi_1 \cdots \pi_{\lfloor N/2 \rfloor}\}$ and σ^N_B as the uniform distribution over $\{\pi_{\lfloor N/2 \rfloor+1} \cdots \pi_N\}$. Define the following two distributions:

$$D^N_1(h) = \mathbb{E}_{\pi \in \sigma^N_A}[T_\pi(h' \mid s,a)]$$

$$D^N_2(h) = \mathbb{E}_{\pi \in \sigma^N_B}[T_\pi(h' \mid s,a)].$$

Define the total variation distance between these two distributions (Billingsley 2012), as

$$\Delta_N = \sum_{h \in \mathcal{H}} \left\| D^N_1(h) - D^N_2(h) \right\|.$$

A perfect detector will return that s, a is child stationary at iteration N if and only if $\Delta_N = 0$.

We assume separate Q-value tables for the stationary and nonstationary updates, so that the updates do not interfere with each other. To determine the policy at each iteration (or episode) n, we chose $\pi^n(a) \propto Q(s,a)$ where Q is the nonstationary Q-table if s, a fails child stationarity and the stationary table otherwise. The result of this is that, with respect to a given s, a, updates done at child stationarity iterations have no impact on updates done at non child stationarity iterations and vice versa.

To prove Theorem 4.4, we proceed by induction. As the base case, consider any terminal infostate s of the game tree where there are no actions. In this case, all policies are no-regret. For the inductive hypothesis, consider any infostate s and assume that ABCs guarantees no-regret for any s', a' where s' is a descendent of s and a' is a valid action at s'. To show is that ABCs also guarantees the no-regret property at s, a. Formally, define $\eta^{\pi}_{-i}(s) = \sum_{h \in H(s)} \eta^{\pi}_{-i}(h)$. We write the *local regret* for a policy π as $r^{\pi}(s,a) = \max\big(0, \eta^{\pi}_{-i}(s)\big(Q^{\pi}(s,a) - \mathbb{E}_{a' \sim \pi(s)}[Q^{\pi}(s,a')]\big)\big)$.

Consider the sequence $(d_n)_n$, where $d_n \in \{0,1\}$ represents whether the oracle reports whether s, a is child stationarity after n iterations (or episodes) of training. There are three cases to consider: $(d_n)_n$ converges to 0 and fails to satisfy child stationarity in the limit, converges to 1 and satisfies child stationarity in the limit, or fails to converge (e.g., perhaps it oscillates between child stationarity and not child stationarity).

Case 1: s, a fails child stationarity in the limit (i.e., $(d_n)_n \to 0$).

In Appendix C, we show that ABCs runs MAX-CFR at every s, a that does not satisfy child stationarity according to the detector. Lemma C.9 shows that running MAX-CFR at a given state and action s, a achieves vanishing local regret at s, a, and the assumption of the perfect oracle means that ABCs will asymptotically converge to MAXCFR at this particular s, a. Combined with our inductive hypothesis, this guarantees that ABCs is no-regret for (s, a) and all its descendent infostate, action pairs.

Case 2: s, a satisfies child stationarity in the limit (i.e., $(d_n)_n \to 1$). Let $\overline{\pi}^N$ denote the average joint policy at episode N. The Multiplicative Weights / Hedge algorithm (Freund and Schapire 1997; Arora, Hazan, and Kale 2012) guarantees that selecting $\pi_{n+1}(s) \propto \text{softmax}\big(Q^{\overline{\pi}_N}(s,*)\big)$ has $\lim_{N\to\infty} r^{\overline{\pi}_N}(s,a) = 0$. When the detector successfully detects s, a as child stationary, Algorithm 2 performs exactly such an update at every infostate visited (except that it uses estimated $\hat{Q}^{\overline{\pi}_N(s,a)}$ values). We also show that the estimated $\hat{Q}^{\overline{\pi}_N}(s,a)$ values converge to the true $Q^{\overline{\pi}_N}(s,a)$ values, resulting in Hedge / Multiplicative weights performing the softmax update on the correct counterfactual regrets. We do so as follows.

Lemma D.1. *Define $G^*(s,a) = \lim_{N\to\infty} G^{\overline{\pi}^N}(s,a)$. If $(d_t)_t \to 1$, then $G^*(s,a)$ is guaranteed to exist.*

Proof. While $G^*(s,a)$ is not a well defined subgame in general, we show that it exists if s, a is asymptotically child stationarity. Considering a vectorized form of the payoff entires in a finite action, normal form game, the set of all games can be written as a subset of $\mathcal{R}^q$, for a suitable q, and forms a compact metric space. $(d_n)_n \to 1$ implies $\Delta_N \to 0$ by construction. Let π_A^N, π_B^N represent the average joint policies as defined in Section 4 at iteration N. $\overline{\pi}^N$ is, by definition, the average joint policy between π_A^N and π_B^N. Using the symmetry of the total variation distance along with the triangle inequality, we have

$$\begin{aligned}
& d\Big(G^{\overline{\pi}^N}(s,a), G^{\overline{\pi}^{N+1}}(s,a)\Big) \\
&= d\Big(G^{\frac{1}{2}\pi_A^N+\pi_B^N}(s,a), G^{\frac{1}{2}\pi_A^{N+1}+\pi_B^{N+1}}(s,a)\Big) \\
&\le d\Big(G^{\frac{1}{2}\pi_A^N+\frac{1}{2}\pi_B^N}(s,a), G^{\pi_A^N}(s,a)\Big) \\
&+ d\Big(G^{\pi_A^{N+1}(s,a)}, G^{\frac{1}{2}\pi_A^{N+1}+\frac{1}{2}\pi_B^{N+1}}(s,a)\Big) \\
&+ d\Big(G^{\pi_A^N}(s,a), G^{\pi_A^{N+1}}(s,a)\Big) \\
&\le \Delta_N + \Delta_{N+1} + \frac{1}{N+1}
\end{aligned}$$

where we have that $d\Big(G^{\frac{1}{2}\pi_A^N+\frac{1}{2}\pi_B^N}(s,a), G^{\pi_A^N}\Big) < d\Big(G^{\pi_B^N}(s,a), G^{\pi_A^N(s,a)}\Big)$ due to the convexity of the total variation distance (Billingsley 2012). Additionally, $d\Big(G^{\pi_A^N}(s,a), G^{\pi_A^{N+1}}(s,a)\Big) \le \frac{1}{N+1}$ because, by construction, π_A^N and π_A^{N+1} are identical in all by at most $\frac{1}{N+1}$ fraction of the joint policies in the support. Since $\lim_{N\to\infty} d\Big(G^{\overline{\pi}^N}(s,a), G^{\overline{\pi}^{N+1}}(s,a)\Big) = 0$, the sequence of games given by $G^{\overline{\pi}^N}(s,a)$ forms a Cauchy sequence and $G^*(s,a)$ is guaranteed to exist since Cauchy sequences converge in compact metric spaces. □

By the assumption that G is a two-player zero-sum game, $G^*(s,a)$ must also be a two-player zero-sum game. By our inductive hypothesis, ABCs has no regret for any s', a' that is a descendent of s, a. This, combined with the existence of $G^*(s,a)$ and (Zinkevich et al. 2008, Theorem 2) means that $\lim_{N\to\infty} \overline{\pi}^N$ converges a Nash equilibrium of $G^*(s,a)$.

The minimax theorem (von Neumann 1928) states that in two-player zero-sum games, players receive the same payoff, known as the minimax value of the game, *regardless of which Nash equilibrium* is discovered. Call this value Q^*. Furthermore, by the assumption that $\lim(d_t)_t \to 1$, the immediate rewards received after taking action a at state s must converge to some value r^*. As such, the estimated Q values must converge to the correct value $\hat{Q}^{\overline{\pi}_N}(s,a) = r^* + \gamma Q^*$, as desired.

Case 3: Imagine that s, a is neither stationary nor nonstationary in the limit. Let

$$A_{nonstat}^N = \{d_t \mid d_t = 0\}, \quad A_{stat}^N = \{d_t \mid d_t = 1\}.$$

Each of $\lim_{N\to\infty} \|A_{stat}^N\|$ and $\lim_{N\to\infty} \|A_{nonstat}^N\|$ must both diverge, since if one of them were only finitely large, then this would contradict the assumption that $(d_t)_t$ does not converge.

As described in our setup, because we track separate Q values for stationary and nonstationary updates and use a perfect oracle, the updates done in iterations A_{stat}^N have no impact on the updates done on $A_{nonstat}^N$ and vice versa. As such, consider the subset of timesteps given by $A_{nonstat}^N$. By direct reduction to Case 1, the distribution over joint policies given by sampling uniformly over $\{\pi_k \mid k \in A_{nonstat}^N\}$ must satisfy the no-regret property for s, a. Analogously, the distribution over joint policies given by sampling uniformly over $\{\pi_k \mid k \in A_{stat}^N\}$ must satisfy the no-regret property for s, a by direct reduction to Case 2.

As the inductive hypothesis is satisfied for both partitions A_{stat}^N and $A_{nonstat}^N$, we can continue to induct independently for each partition. The induction terminates at the initial infostate s_0 of our game because we assume finite depth. Suppose we are left at this step with K nonoverlapping partitions of $\{1 \cdots N\}$. Denote them $A_1^N \cdots A_K^N$, and analogously define $\pi_N^k = \frac{1}{\|A_k^N\|}\sum_{n\in A_k^N} \pi_n$. By our inductive hypothesis and the minimax theorem, we are guaranteed that $\lim_{N\to\infty} r^{\pi_N^k}(s,a) = 0$, for all s, a, k. Since $\overline{\pi}^N = \frac{1}{N}\sum_k \|A_k^N\| \pi_N^k$, we can write

$$\begin{aligned}
\max_a r^{\overline{\pi}^N}(s,a) &= \max_a \frac{1}{N}\sum_k \|A_k^N\| r^{\pi_N^k}(s,a) \\
&\le \frac{1}{N}\sum_k \|A_k^N\| \max_a r^{\pi_N^k}(s,a).
\end{aligned}$$

Lastly, since $\lim_{N\to\infty} r^{\pi_N^k}(s,a) = 0$ for all s, a, k, it must also be true that $r^{\overline{\pi}^N}(s,a) \to 0$ as desired.

□

Theorem 4.5. *Given an appropriate choice of significance levels α, Algorithm 2 asymptotically converges to the optimal policy with only a worst-case $O(A)$ factor slowdown compared to BQL in an MDP.*

Proof. We choose the significance levels α across different infostates as follows. As described in Algorithm 2, at each infostate s encountered, ABCs chooses a single action a_{traj} and branches this, regardless of whether s, a_{traj} satisfies child stationarity with respect to the policies encountered during the learning procedure. Define the set of infostates $s_0 \cdots s_t$ where s_0 is the initial infostate and s_k is the infostate observed after playing the corresponding a_{traj} action at s_{k-1}. This "trajectory" is the set of infostates that ABCs would have updated even if the environment was fully child stationary and the direct analogue of the trajectory that BQL would have updated. Note that the chosen "trajectory" may change during every iteration of the learning procedure.

For any infostate on this constructed trajectory, set $\alpha \le \frac{1}{A^d}$, where A is the maximum number of actions at any infostate and d is the depth of infostate s. At all other infostates, set $\alpha \le \frac{1}{A}$.

This choice of significance levels guarantees only an $O(A)$ factor slowdown compared to BQL. Let $X_{ABCs}(d)$ count the number of infostates of depth d that ABCs updates at each iteration.Since the environment is an MDP, an infostate is updated only if the detection algorithm hit a false positive on every ancestor of the infostate that was not on the trajectory path. We bound this probability as follows. Consider an infostate s with depth d. Let d^* be the highest depth infostate in the ancestry of s that was on the "trajectory" path. As false positives asymptotically occur with probability α in a Chi-Squared test (Pearson 1900) [6],the probability that s is visited approaches,

$$\frac{1}{d^*}\prod_{i=d^*+1}^{d}\frac{1}{A}=A^{-d}.$$

There are only at most A^d nodes at depth d in the tree, and thus, in expectation, we can expect at most $A^{-d}A^d = O(1)$ extra updates at level d in the tree, or $O(D)$ total extra updates for a game tree of depth D. Since MAXCFR is a contraction operator over the potential function which bounds the regret (Appendix C, additional, unnecessary updates at infostates falsely deemed to be nonstationary by the detector in Algorithm 1 will not hurt the asymptotic convergence rate, as MAXCFR also converges the optimal policy of an MDP.

Since our environment is noncyclical and BQL learns from trajectories, ABCs will do $O(D)$ updates on a game tree of depth D, completing the proof. However, ABCs updates every action at an infostate whereas BQL only updates a single action, giving the additional $O(A)$ inefficiency relative to BQL. □

E Experimental Details

Testing Environments

We present more detailed descriptions of our testing environments, including any modifications made to their standard implementations, below.

Cartpole Cartpole is a classic control game from the OpenAI Gym, in which the agent must work to keep the pole atop the cart upright while keeping the cart within the boundaries of the game. While Cartpole has a continuous state space, represented by a tuple containing (cart position, cart velocity, pole angle, pole angular velocity), we discretize the state space for the tabular setting as follows:

- **Cart position:** 10 equally-spaced bins in the range $[-2.4, 2.4]$
- **Cart velocity:** 10 equally-spaced bins in the range $[-3.0, 3.0]$
- **Pole angle:** 10 equally-spaced bins in the range $[-0.5, 0.5]$
- **Pole angular velocity:** 10 equally-spaced bins in the range $[-2.0, 2.0]$

Additionally, to make convergence feasible, we parameterize the game such that the infostates exactly correspond to the discretized states, rather than the full sequence of states seen and actions taken. Note that this means that our version of Cartpole does not satisfy perfect recall; to account for this and ensure that we are not repeatedly branching the same infostate, we prevent ABCs and MAX-CFR from performing a CFR exploratory update on the same infoset twice in a given iteration.

The standard implementation of Cartpole is technically non-Markovian as each episode is limited to a finite number of steps (200 in the v0 implementation). To make the environment Markovian, we allow the episode to run infinitely, but we introduce a $1/200$ probability of terminating at any given round, thus ensuring that the maximum expected length of an episode is precisely 200.

Weighted Rock-Paper-Scissors Weighted rock-paper-scissors is a slight modification on classic rock-paper-scissors, where winning with Rock yields a reward of 2 whereas winning with any other move only yields a reward of 1 (a draw results in a reward of 0). The game is played sequentially, but player 2 does not have knowledge of the move player 1 makes beforehand (possible via use of imperfect-information / infostates).

Kuhn Poker Kuhn Poker is a simplified version of poker with only three cards – a Jack, Queen, and King. At the start of the round, both players receive a card (with no duplicates allowed). Each player antes 1 – player 1 then has the choice to bet or check an amount of 1. If player 1 bets, player 2 can either call (with both players then revealing their cards) or fold. If player 1 checks, player 2 can either raise an amount of 1 or check (ending the game). In the event that player 2 raises, player 1 then has the choice to call or fold, with the game terminating in either case.

We use the standard OpenSpiel implementation of Kuhn poker for all our experiments.

Leduc Poker Leduc poker is another simplified variant of poker, though it has a deeper game tree than Kuhn poker. As with Leduc, there are three cards, though there are now two suits. Each player receives a single card, and there is an ante of 1 along with two betting rounds. Players can call, check, and raise, with a maximum of two raises per round. Raises in the first round are two chips while they are four chips in the second round.

Again, we use the standard OpenSpiel implementation of Leduc poker for all our experiments.

Cartpole + Leduc Poker For our stacked environment, player 1 first plays a round of Cartpole. Upon termination of this round of Cartpole, they then play a single round of Leduc poker against player 2. Note that player 2 only ever interacts with the Leduc poker environment.

We use the same implementations of Cartpole and Leduc poker discussed above, with the sole change being that we set Cartpole's termination probability to $1/100$ rather than $1/200$ to feasibly run MAX-CFR on the stacked environment.

Hyperparameters Used

For all of the experiments in the main section of the paper, we use the same hyperparameters for all of the algorithms tested. We enumerate the hyperparameters used for BQL, MAX-CFR, and the MCCFR methods in Table 1.

[6] Since the Chi-Squared test is an asymptotic test, this is true in the limit, but not necessarily for finite samples.

We tuned the hyperparameters for MCCFR methods, BQL, and ABCs as to optimize performance over all five environments for which experiments were conducted.

Table 1: Benchmark Hyperparameter Values

Hyperparameters	Final Value
Discount Factor	$\gamma = 1$
BQL Temperature Schedule	$\tau_n = 10 \cdot (0.99)^{\lfloor n/50 \rfloor}$
MAX-CFR Temperature Schedule	$\tau_n = 1$
OS-MCCFR ϵ-greedy Policy	$\epsilon = 0.6$

For ABCs, we use the following set of hyperparameters, enumerated in Table 2. As noted in the main paper, we make a number of practical modifications to Algorithm 2. In particular, we use a fixed p-value threshold, allow for different temperature schedules for stationary/nonstationary infosets, choose not to use ϵ-exploration, and only perform the stationarity check with some relatively small probability.

Table 2: ABCs Hyperparameter Values

Hyperparameters	Final Value
Discount Factor	$\gamma = 1$
Nonstationary Temperature Schedule	$\tau_n = 1$
Stationary Temperature Schedule	$\tau_n = (0.99)^{\lfloor n/20 \rfloor}$
Cutoff p-value	$\alpha_s = 0.05$
ϵ-exploration	$\epsilon = 0$
Probability of Stationarity Check	$p_{check} = 0.05$

Evaluation Metrics

For all our experiments, we plot "nodes touched" on the x-axis, a standard proxy for wall clock time that is independent of any hardware or implementation details. Specifically, it counts the number of nodes traversed in the game tree throughout the learning procedure. On each environment, we run all algorithms for a fixed number of nodes touched, as seen in the corresponding figures.

For all multi-agent settings, we plot exploitability on the y-axis. For Cartpole, we plot regret, which is simply the difference between the reward obtained by the current policy and that obtained by the optimal policy (200 minus the current reward).

Random Seeds

We use three sequential random seeds $\{0, 1, 2\}$ for each experiment.

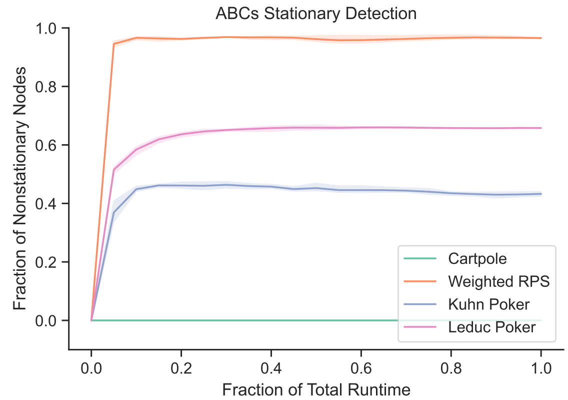


Figure 2: Stationarity detection for Cartpole, RPS, Leduc poker, and Kuhn poker. The fraction of infostates detected as non-stationary is plotted over time.

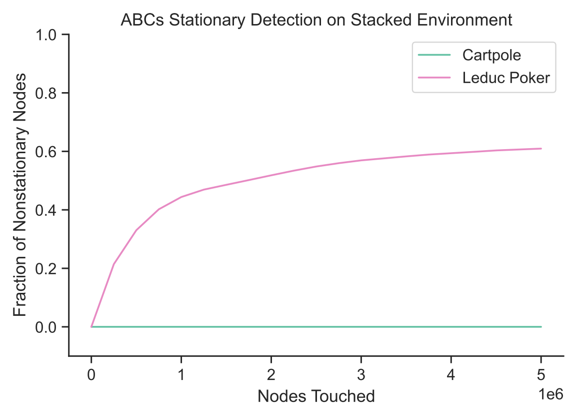


Figure 3: Nonstationary detection on stacked Cartpole/Leduc poker environment.

F Additional Experiments

Stationarity Detection

We plot percentage of infostates detected as nonstationary over time for each of the non-stacked environments in Figure 2. We normalize the x-axis to plot the fraction of total runtime/nodes touched so that all environments can be more easily compared.

Note that the multi-agent environments possess varying levels of stationarity. The policies in weighted RPS are cyclical, meaning that the corresponding transition functions change throughout the learning procedure. In contrast, in our two poker environments, there are several transition functions that may stay fixed, either because the outcome of that round is determined by the cards drawn or because agents' policies become fixed over time.

To take an example, in the Nash equilibrium of Kuhn poker, player 1 only ever bets in the opening round with a Jack or a King. Thus, if player 2 is in the position to call with a King, betting will necessarily yield the history in which players have cards (K, J) and player 2 calls player 1's initial bet.

To better visualize the performance of ABCs on our stacked Cartpole/Leduc environment, we plot the fraction of infostates detected as nonstationary in each environment separately in Figure 3.

TicTacToe

We also run experiments on TicTacToe. While TicTacToe is not a fully stationary game – as it is still a multi-agent setting – it is a perfect information game, meaning that BQL should still be able to learn the optimal policy of the game. As with Cartpole, we modify the infostates to depend only on the current state of the board.

We plot results for ABCs and benchmarks in Figure 4, with exploitability clipped at 10^{-5} to facilitate easier comparison. To accelerate convergence, we evaluate the current greedy policy rather than the average policy – note that last-iterate convergence implies average policy convergence. We additionally modify the temperature schedules for ABCs and BQL to accommodate the larger game tree. These are listed in Table 3.

Table 3: TicTacToe Hyperparameter Values

Hyperparameters	Final Value
ABCs Non-stationary Temperature Schedule	$\tau_n = 1$
ABCs Stationary Temperature Schedule	$\tau_n = 10 \cdot (0.99)^{\lfloor n/50 \rfloor}$
BQL Temperature Schedule	$\tau_n = 10 \cdot (0.99)^{\lfloor n/100 \rfloor}$

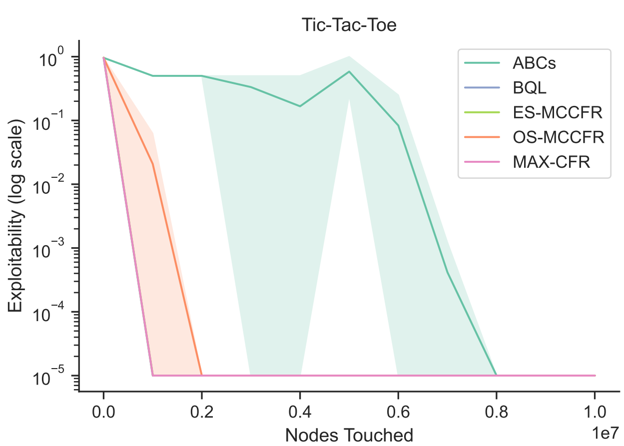


Figure 4: Performance on TicTacToe, exploitability is clipped at 10^{-5} and plotted on log scale.

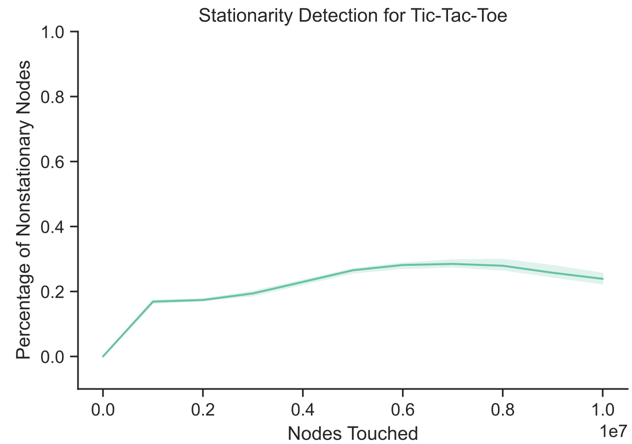


Figure 5: Nonstationary detection on TicTacToe.

G Additional Related Work

We expand on the related work from Section 1.

Methods from Game Theory

While CFR guarantees polynomial time convergence in MDPs (assuming the MDP satisfies perfect recall), empirically it performs far slower than its corresponding reinforcement learning counterparts despite the same worst case bound. This is primarily because, in practice, reinforcement learning methods such as PPO (Schulman et al. 2017) or DQN (Mnih et al. 2015) are able to learn reasonable policies in benchmarks such as the Atari Learning Environment (Bellemare et al. 2013) while exploring only a tiny fraction of the possible states in the environment. Meanwhile, CFR requires updating every infostate in the entire game tree at every iteration of the game.

Monte Carlo Counterfactual Regret Minimization External sampling Monte-Carlo CFR (ES-MCCFR) and Outcome sampling Monte-Carlo CFR (OS-MCCFR) (Lanctot et al. 2009) are two methods that have been proposed to alleviate this issue. ES-MCCFR samples a single action for each player other than the player conducting the learning procedure (including the chance player) for each iteration, resulting in substantially fewer updates compared to the full version of CFR. However, because external sampling still requires performing a single update at every infostate of every player, it still requires $O(A^D)$ updates per iteration, where A is the number of actions and D is the maximum depth of the game tree, with respect to any single player. OS-MCCFR is similar to ES-MCCFR except that it only samples a single trajectory and then corrects its estimates of the counterfactual values using importance sampling. While this update is indeed a trajectory style update like Q-learning, the variances of the estimated values are exceedingly high as they involve dividing by the reach probability of reaching an infostate, which will be on the order of $O(A^{-D})$. As such, OS-MCCFR will still require on the order of $O(A^D)$ trajectories before it learns a reasonable policy.

Several methods have been proposed to empirically improve the asymptotic convergence rate with respect to the number of iterations. CFR+ (Tammelin 2014) is a popular method that modifies the regret estimate to be an upper bound of the actual regrets. The same work also introduced alternating updates, which is only updating the regrets of each player every P iterations, where P is the number of players in the game. Linear CFR (Brown and Sandholm 2019a) and its follow up work greedy weights (Zhang, Lerer, and Brown 2022) modify the weighting scheme of the CFR updates. All the above methods empirically improve the convergence rate but maintain the worst case bound.

Methods from Reinforcement Learning

While reinforcement learning was designed for finding optimal policies on MDPs, many attempts have been made to adapt such algorithms to the multi-agent setting. Algorithms such as Q-learning and PPO are generally not guaranteed to converge to equilibria in such settings (Brown and Sandholm 2019a).

AlphaStar AlphaStar (Vinyals et al. 2019) attempted to learn the real-time strategy game of Starcraft via reinforcement learning. Since Starcraft is a two-player imperfect information game, to combat the nonstationarity involved in playing multiple different opponents who required different strategies to defeat, AlphaStar trained against a league of agents instead of a single agent as is usually the case in self play. Such a league simulates the distribution of human adversaries AlphaStar is likely to encounter while playing on the Blizzard Starcraft ladder. While this method performed reasonable well in practice, it did not come with guarantees of convergence to the Nash equilibria of the game.

OpenAI Five OpenAI Five (OpenAI 2019) was a similar project to learn DOTA 2, a popular multiplayer online battle arena video game. In order to combat the nonstationarity, OpenAI Five played in a restricted version of DOTA 2, in which aspects of the gave that involved imperfect information (such as wards which granted vision of enemy areas) were removed from the game. Indeed, while the resulting agent was able to beat some of the best human teams in the world online players quickly found strategies that coule exploit the agents.

RMAX On the more theoretical side, RMAX (Brafman and Tennenholtz 2002) is a classic algorithm from reinforcement learning that achieves both optimal reward in an MDP and the minimax value of a stochastic game in polynomial time. RMAX operates by maintaining an optimistic estimate of the Q-values of each state which upper bounds the possible value of each state and thus encourages exploration of such states. This optimistic upper bound ensures However, there are several differences between RMAX and ABCs. For starters, RMAX is not model free – it requires a model of the environment to estimate the Q values. Additionally, unlike CFR (and by extension ABCs), it is not Hannan consistent, meaning it will not play optimally against an arbitrary adversary in the limit. On a more practical level, RMAX leads to substantially more exploration than its related methods such as Q-learning by design, and as a result is used less often in practice compared to variants such as Q-learning which often are able to learn reasonable policies while exploring only a tiny fraction of the environment.

Nash Q-Learning Nash Q-Learning (Hu and Wellman 2003) is an adaptation of the standard Q-learning algorithm finding the minimax value of stochastic game, which corresponds to the Nash equilibrium in two-player zero-sum games. Instead of doing a standard Q-learning update, Nash Q-learning calculates the Nash equilibrium of the subgame implied by the Q-values of all players and performs an update according to that policy, instead of the policy being currently followed by all players. However, there are several important differences between ABCs and Nash Q-learning. Firstly, in our multi-agent learning setup, ABCs is run for each player of the game, but the learning algorithms for each player are run separately, with no interaction between players except via the game itself. In contrast, Nash-Q requires a centralized setup in which players collectively choose their individual strategies as a function over the Q-values over *all players in the game*. In this sense, Nash Q-learning is a collective algorithm for finding equilibria of the game as opposed to a learning algorithm trying to optimize the reward of each agent. Additionally, Nash Q-learning requires a Nash equilibrium solver for every update of the Q-values. While this is possible in polynomial time in a two-player zero-sum game, it is more expensive than the analogous update in ABCs and Q-learning which is done in constant time.

Unified Methods from Game Theory and Reinforcement Learning

Magnetic Mirror Descent Magnetic Mirror Descent (MMD) (Sokota et al. 2023) is similarly a unified algorithm capable of solving for equilibria in two-player zero-sum games and in single player settings. However, unlike ABCs, MMD does not adaptively determine whether or not to branch infostates based on whether or not they are stationary. As such, it cannot simultaneously guarantee convergence to Nash in two-player zero-sum games and matching performance against BQL with the same set of hyperparameters. MMD runs experiments on a variety of single and multi-agent environments, but requires different hyperparameters for each one, most importantly those governing how many infostates to explore. In contrast, ABCs manages to roughly match the performance of Boltzmann Q-Learning in stationary environments and the performance of counterfactual regret minimization —— all with the same set of hyperparameters due to adaptively choosing how much of the tree to explore based on how stationary the infostate is.

Player of Games Player of Games (Schmid et al. 2021) is an algorithm that is able to simultaneously achieve superhuman status on many perfect and imperfect information games by combining lessons from both AlphaZero and counterfactual regret minimization. Instead of exploring the whole game tree as is the case with standard CFR, Player of Games expands a fixed number of infostates at each timestep. This is in contrast to ABCs which performs an adaptive choice on how many children to expand depending on whether the infostate and action satisfies child stationarity. As a result, PoG performs asymptotically slower than its counterparts designed for perfect information / single agent environments as a price for its generality. In contrast, we show that ABCs provably has only a constant factor slowdown compared to its reinforcement learning counterpart in the limit case for a stationary environment by converging to updating the same set of infostates as BQL.