Faster Game Solving via Hyperparameter Schedules

Naifeng Zhang[1], **Stephen Marcus McAleer**[2], **Tuomas Sandholm**[1,3,4,5]
[1]Carnegie Mellon University
[2]Anthropic
[3]Strategy Robot, Inc.
[4]Strategic Machine, Inc.
[5]Optimized Markets, Inc.
naifengz@cmu.edu, mcaleer.stephen@gmail.com, sandholm@cs.cmu.edu

Abstract

Counterfactual regret minimization (CFR) algorithms are a foundational class of methods for solving imperfect-information games, with the time average of their iterates converging to a Nash equilibrium in two-player zero-sum games. Prior state-of-the-art variants, *Discounted CFR (DCFR)* and *Predictive CFR$^+$ (PCFR$^+$)*, achieved the fastest known practical performance by improving convergence rates over vanilla CFR through discounting early iterations with a fixed discounting scheme. More recently, *Dynamic DCFR (DDCFR)* introduced agent-learned dynamic discounting schemes to further accelerate convergence, at the cost of increased complexity. To address this, we propose *Hyperparameter Schedules (HSs)*, a remarkably simple, training-free framework that dynamically adjusts CFR discounting over time. HSs aggressively downweight early updates and gradually transition to trusting late-stage strategies, leading to substantially faster convergence with only a few lines of code modifications. We show that HSs derived from just three small extensive-form games generalize effectively to 17 diverse games (including large-scale realistic poker) in both extensive-form and normal-form settings, without any game-specific tuning. Our method establishes a new state of the art for solving two-player zero-sum games.

Extended version — https://arxiv.org/pdf/2404.09097

1 Introduction

Most real-world settings are *imperfect-information games (IIGs)*, where players hold private information and there may also be information that no player knows. To perform well in IIGs, players typically randomize their strategies to avoid being too predictable and to prevent revealing too much of their private information. In many IIGs, this amounts to understanding deception and being able to deceive, challenges that do not arise in perfect-information games such as chess or Go. Over the past two decades, there have been tremendous breakthroughs in techniques for solving imperfect-information games. Like most of the literature, we focus on solving two-player zero-sum IIGs by converging to a Nash equilibrium (Nash 1950), in which no player can improve by deviating from the equilibrium. The most popular computational approach for converging to a Nash equilibrium in IIGs is the family of *counterfactual regret minimization (CFR)* algorithms (Zinkevich et al. 2007). These methods systematically reduce the regrets of both players through an iterative process, progressively guiding the time-averaged strategy of each player toward a Nash equilibrium (minmax) strategy. CFR-type algorithms serve as the cornerstone for various *state-of-the-art (SoTA)* algorithms in the field (Bowling et al. 2015; Moravčík et al. 2017; Brown and Sandholm 2018, 2019a,b; Zarick et al. 2020; Farina, Kroer, and Sandholm 2021; McAleer et al. 2023; Xu et al. 2024).

Several innovative improvements to CFR exhibit substantially faster convergence. The introduction of CFR$^+$ (Tammelin 2014) marked a major milestone, demonstrating an order of magnitude better convergence rate than vanilla CFR in practice. Specifically, CFR$^+$ 1) alternates strategy updates between players, 2) uses *regret-matching$^+$ (RM$^+$)* instead of *regret matching (RM)* as its regret minimizer, and 3) implements a linear discounting scheme where the contribution of iteration t to the average strategy is weighted by t. This advancement played a pivotal role in almost optimally solving the two-player limit Texas hold 'em poker (Bowling et al. 2015). Building on the idea of discounting, *Discounted CFR (DCFR)* introduces a family of algorithms that, prior to the present paper, represented the practical SoTA for poker games and other games with poker-like structure (Brown and Sandholm 2019a). Specifically, DCFR incorporates discounts for both regrets and the average strategy by assigning more weight to later iterations using three hyperparameters: α, β, and γ. More recently, *Predictive CFR$^+$ (PCFR$^+$)* (Farina, Kroer, and Sandholm 2021) stands out as an effective approach that utilizes predictive Blackwell approachability. Prior to the present paper, PCFR$^+$ was the SoTA for zero-sum games other than poker.

Despite the faster convergence achieved by fixed discounting schemes in these CFR variants, recent work shows that dynamically adjusting the discounting scheme across iterations can further accelerate convergence. Greedy Weights (Zhang, Lerer, and Brown 2022) was the first proposed regret minimization algorithm that dynamically adjusts iteration weights based on runtime observations. The method was primarily focused on approximating equilibria in normal-form games and exhibited poor performance when applied to extensive-form games (Xu et al. 2024). *Dynamic DCFR (DDCFR)* (Xu et al. 2024), a recent innova-

tion, uses a *reinforcement learning (RL)* framework trained on four small games to acquire a well-performing discounting scheme that dynamically adjusts the hyperparameters of DCFR at runtime. DDCFR has demonstrated its effectiveness across several games, leveraging the knowledge gained from the training process and game-specific inference. However, the approach incurs additional computation costs from feature calculations, policy training, and network inference for real-time computation of discounting weights. It also assumes that the hyperparameter-changing policy learned in the training games generalizes well to the target games.

In this work, we investigate whether a lightweight and training-free approach can match or even surpass prior SoTA performance by leveraging the benefits of a dynamic discounting scheme without incurring the complexity and computational overhead of training and running an RL agent. We build on an established insight that, within the CFR family of algorithms, the computed strategy becomes increasingly refined over time, making it reasonable to assign greater weight to more recent iterations. While this idea has been explored in earlier work, we show that the new schemes introduced in this work are significantly more effective than existing ones in the literature. We introduce the concept of *Hyperparameter Schedules (HSs)*—schemes that govern how the hyperparameters of the underlying equilibrium-finding algorithm change across iterations. We then propose two performant HSs that, unlike prior work, *aggressively* discount early iterations and *gradually* increase the contribution of later updates as CFR progresses. Specifically, we augment DCFR with schedules for its hyperparameters (α, β, and γ) using fewer than 15 lines of code changes, resulting in the algorithm HS-DCFR. Similarly, we enhance PCFR$^+$ by applying a schedule to its hyperparameter γ with fewer than 10 lines of code changes, resulting in the algorithm HS-PCFR$^+$. We prove that both algorithms converge to a Nash equilibrium at the stated worst-case rates, provided that the hyperparameters of the HS stay within certain ranges. We show via extensive experiments that the new algorithms yield better solutions for a given amount of run time—*in many games by many orders of magnitude without any game-specific tuning*. These algorithms constitute the new SoTA for both extensive-form and normal-form zero-sum games.

2 Background

In this section, we introduce the preliminaries of extensive-form (i.e., tree-form) games, which constitute the primary subject of our study. Our notation and presentation follow established conventions in prior work (Brown and Sandholm 2015, 2019a; Xu et al. 2024).

2.1 Notation

Let us denote the finite set of players in an extensive-form game (Osborne and Rubinstein 1994) as $\mathcal{N}$, along with a unique player called *chance*, denoted by c. Chance has a fixed, known stochastic strategy. A *history* h contains all information at the current situation. The set of all histories in the game tree is denoted by $\mathcal{H}$. $\mathcal{A}(h)$ denotes the *actions* available at a given history h, and $\mathcal{P}(h)$ is the unique *player* whose turn it is to act at history h. We write $h \cdot a = h'$ if action $a \in \mathcal{A}(h)$ leads from history h to another history h'. At *terminal histories* $\mathcal{Z} \subseteq \mathcal{H}$, no actions are available. For terminal history $z \in \mathcal{Z}$, each player $i \in \mathcal{N}$ has a utility function $u_i(z) : \mathcal{Z} \to \mathbb{R}$. Let us denote the *range of utilities* reachable by player i as Δ_i. Formally, $\Delta_i = \max_{z\in\mathcal{Z}} u_i(z) - \min_{z\in\mathcal{Z}} u_i(z)$ and $\Delta = \max_{i\in\mathcal{N}} \Delta_i$.

Players not having complete knowledge regarding the game state is represented by *information sets* $\mathcal{I}_i$ for each player $i \in \mathcal{N}$. Any history h, h' in the same information set $I \in \mathcal{I}_i$ are indistinguishable to player i. Every non-terminal history $h \in \mathcal{H}$ belongs to exactly one I for each player. We define $\mathcal{A}(I)$ as the set of actions such that $\forall h \in I, \mathcal{A}(I) = \mathcal{A}(h)$. Let $|\mathcal{A}_i| = \max_{I\in\mathcal{I}_i} |\mathcal{A}(I)|$, $|\mathcal{A}| = \max_i |\mathcal{A}_i|$, and $|\mathcal{I}| = \sum_{i\in\mathcal{N}} |\mathcal{I}_i|$.

For player i, a *strategy* $\sigma_i(I)$ is a probability distribution on the actions available in each information set I. The probability distribution of an action a selected by player i is denoted by $\sigma_i(I, a)$. Given that all histories within an information set I are indistinguishable for player i, strategies for all histories within I are identical. Thus, we define $\sigma_i(h, a) = \sigma_i(I, a)$ for any $h \in I$ and $i = \mathcal{P}(h)$. Let σ_i denote a comprehensive strategy for player i, which specifies the actions player i chooses in each information set $I \in \mathcal{I}_i$. We define σ_{-i} as the strategies of all players other than player i. $u_i(\sigma_i, \sigma_{-i})$ is the *expected utility* for player i if player i follows σ_i and all other players follow σ_{-i}.

Let σ denote a *strategy profile* that contains a strategy σ_i for each player i. Let $g \sqsubseteq h$ denote that g is either equal to h or g is a *prefix* of h in the tree structure. The *history reach probability* of a history h, defined as $\pi^\sigma(h) = \prod_{h'\cdot a\sqsubseteq h} \sigma_{\mathcal{P}(h')}(h', a)$, represents the joint probability of reaching h if all players play according to σ. The contribution of player i to the history reach probability is defined as $\pi_i^\sigma(h)$ and the contribution of all other players is defined as $\pi_{-i}^\sigma(h)$. The *information set reach probability* $\pi^\sigma(I)$ is defined as $\sum_{h\in I} \pi^\sigma(h)$. The *interval history reach probability* from history h to h' is expressed as $\pi^\sigma(h, h') = \pi^\sigma(h')/\pi^\sigma(h)$ if $h \sqsubseteq h'$.

2.2 Nash Equilibrium

The *best response* to a strategy σ_{-i} is any strategy $\mathrm{BR}(\sigma_{-i})$ that maximizes the expected utility for player i, such that $u_i(\mathrm{BR}(\sigma_{-i}), \sigma_{-i}) = \max_{\sigma_i'} u_i(\sigma_i', \sigma_{-i})$. A *Nash equilibrium* (Nash 1950) is a strategy profile $\sigma^* = (\sigma_i^*, \sigma_{-i}^*)$ where each player adopts a strategy that serves as the best response to the strategies chosen by the other players. Formally, $\forall i \in \mathcal{N}, u_i(\sigma_i^*, \sigma_{-i}^*) = \max_{\sigma_i'} u_i(\sigma_i', \sigma_{-i}^*)$. The *exploitability* of a strategy σ_i measures how far it is from a Nash equilibrium in the sense of utility: $e(\sigma_i) = u_i(\sigma_i^*, \mathrm{BR}(\sigma_i^*)) - u_i(\sigma_i, \mathrm{BR}(\sigma_i))$. In an *$\epsilon$-Nash equilibrium*, no player has exploitability higher than ϵ. Let $e(\sigma)$ denote the exploitability of a strategy profile. We can compute $e(\sigma)$ using the formula: $e(\sigma) = \sum_{i\in\mathcal{N}} e(\sigma_i)/|\mathcal{N}|$. This is a measure of how far, strategically speaking, a strategy profile is from a Nash equilibrium.

2.3 Counterfactual Regret Minimization

CFR systematically refines players' average strategies by minimizing their regrets over iterations, serving as a prominent equilibrium-finding algorithm for extensive-form IIGs (Zinkevich et al. 2007). In CFR, the *counterfactual value* represents the expected utility of a given information set I for player i assuming that player i attempts to reach I. Formally, given strategy profile σ, the counterfactual value for player i at an information set $I \in \mathcal{I}_i$ is defined as $v_i^\sigma(I) = \sum_{h \in I}(\pi^\sigma_{-i}(h) \sum_{z \in \mathcal{Z}}(\pi^\sigma(h,z)u_i(z)))$. Similarly, the counterfactual value for player i with action a at I is defined as $v_i^\sigma(I,a) = \sum_{h \in I}(\pi^\sigma_{-i}(h) \sum_{z \in \mathcal{Z}}(\pi^\sigma(h \cdot a, z)u_i(z)))$. Let σ^t be the strategy profile on iteration t. The *instantaneous regret* on iteration t for player i for not choosing action a in I is defined as $r_i^t(I,a) = v_i^{\sigma^t}(I,a) - v_i^{\sigma^t}(I)$. The *cumulative regret* on iteration T for player i of not choosing action a in I is then defined as $R_i^T(I,a) = \sum_{t=1}^T r_i^t(I,a)$. We also define $R_i^{T,+}(I,a) = \max(R_i^T(I,a), 0)$ and $R_i^T(I) = \max_a(R_i^{T,+}(I,a))$.

CFR typically uses *regret matching (RM)* as the regret minimizer due to its simplicity and lack of parameters (Hart and Mas-Colell 2000; Cesa-Bianchi and Lugosi 2006). In RM, a player chooses actions in each information set according to a distribution proportional to the positive regret associated with those actions. Formally, on iteration $T+1$, player i chooses actions $a \in \mathcal{A}(I)$ according to the strategy

$$\sigma_i^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a' \in \mathcal{A}(I)} R_i^{T,+}(I,a')}, & \text{if } \sum_{a'} R_i^{T,+}(I,a') > 0, \\ \frac{1}{|\mathcal{A}(I)|}, & \text{otherwise.} \end{cases} \quad (1)$$

The *cumulative strategy* on iteration T for player i with action a in an information set I is defined as $C_i^T(I,a) = \sum_{t=1}^T(\pi_i^{\sigma^t}(I)\sigma_i^t(I,a))$. If player i employs CFR on each iteration, then on iteration T, the regret for an information set I satisfies $R_i^T(I) \leq \Delta_i \sqrt{|\mathcal{A}(I)|}\sqrt{T}$. The regret for player i in the entire game satisfies that $R_i^T \leq \sum_{I \in \mathcal{I}_i} R_i^T(I) \leq |\mathcal{I}_i|\,\Delta_i \sqrt{|\mathcal{A}_i|}\sqrt{T}$. Thus, as $T \to \infty$, $R_i^T/T \to 0$. In two-player zero-sum IIGs, if the average regret R_i^T/T of both players is less than or equal to ϵ, their average strategies constitute a 2ϵ-equilibrium (Waugh et al. 2009). Consequently, CFR serves as an anytime algorithm capable of finding an ϵ-Nash equilibrium in zero-sum games.

2.4 Discounted CFR Variants

Numerous improvements to CFR have been introduced that significantly enhance convergence speed. Due to space constraints, in this section, we provide only a brief overview of the SoTA discounted CFR algorithms and refer readers to the cited works for more details.

Building on vanilla CFR, CFR$^+$ (Tammelin 2014) improves convergence rate by an order of magnitude in practice via incorporating alternating updates, using a variation of RM, and applying a linear discounting scheme when computing the average strategy. *Discounted CFR (DCFR)* (Brown and Sandholm 2019a) comprises a family of algorithms that apply discounts to both cumulative regrets and the average strategy. Although DCFR has a theoretical convergence bound that is worse than that of CFR by a constant factor, it has been shown to converge much faster in practice. DCFR's discounting scheme is parameterized by three hyperparameters: α, β, and γ. On iteration $t+1$, DCFR multiplies positive cumulative regrets by $\frac{t^\alpha}{t^\alpha+1}$, negative cumulative regrets by $\frac{t^\beta}{t^\beta+1}$, and contributions to the average strategy by $(\frac{t}{t+1})^\gamma$. Formally, for player i,

$$R_i^{t+1}(I,a) = \begin{cases} R_i^t(I,a)\frac{t^\alpha}{t^\alpha+1} + r_i^{t+1}(I,a), & \text{if } R_i^t(I,a) > 0, \\ R_i^t(I,a)\frac{t^\beta}{t^\beta+1} + r_i^{t+1}(I,a), & \text{otherwise, and} \end{cases} \quad (2)$$

$$C_i^{t+1}(I,a) = C_i^t(I,a)(\tfrac{t}{t+1})^\gamma + \pi_i^{\sigma^{t+1}}(I)\sigma_i^{t+1}(I,a). \quad (3)$$

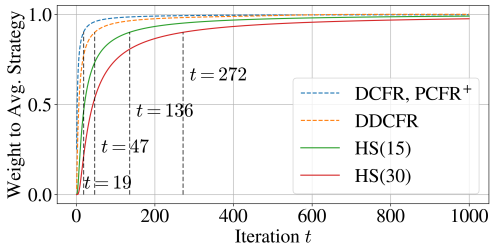
The authors of DCFR recommend setting (α, β, γ) to $(1.5, 0, 2)$, as this configuration consistently outperforms CFR$^+$ in their experiments.

Recently, researchers introduced *Dynamic DCFR (DDCFR)* (Xu et al. 2024) that employs an RL framework. This framework encapsulates CFR's iteration process within an environment, treating the discounting scheme as an agent interacting with it. The RL agent within DDCFR is trained on four small games and, for each test game, DDCFR performs inference on how to dynamically change the hyperparameters (α, β, γ) of DCFR across iterations. *Predictive CFR$^+$ (PCFR$^+$)* (Farina, Kroer, and Sandholm 2021) extends Blackwell approachability (Blackwell 1956) to regret minimization to form *predictive regret matching (PRM+)*. PRM+ uses the observed utility in each iteration during the run as the prediction of the next utility. Additionally, PCFR$^+$ employs a quadratic discounting scheme that can be formalized in the same way as in DCFR (i.e., Equation 3) by setting γ to 2. The authors of DDCFR (Xu et al. 2024) also use the RL framework to learn a dynamic discounting scheme for PCFR$^+$, resulting in *Dynamic PCFR$^+$ (DPCFR$^+$)*. PCFR$^+$ and DPCFR$^+$ are the prior practical SoTA for non-poker games, while in poker games, DCFR and DDCFR serve as the prior practical SoTA.

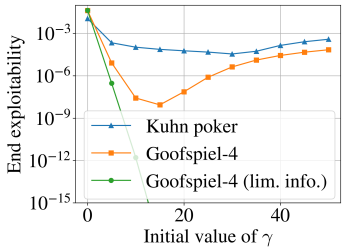
3 Hyperparameter Schedules (HSs)

Many discounted CFR variants, such as CFR$^+$, DCFR, and PCFR$^+$, have demonstrated remarkable performance. However, they rely on a fixed discounting scheme, using the same hyperparameters across all iterations. For example, the authors of DCFR empirically determined that the best DCFR hyperparameters (α, β, γ) are $(1.5, 0, 2)$. Similarly, PCFR$^+$ fixes its discounting scheme by setting γ to 2 across all iterations. In contrast, DDCFR proposes dynamically adjusting hyperparameters using an RL-based framework that trains an agent to determine a dynamic discounting scheme. Specifically, the DDCFR framework learns three HSs that vary each parameter non-linearly within the range $[-5, 5]$.

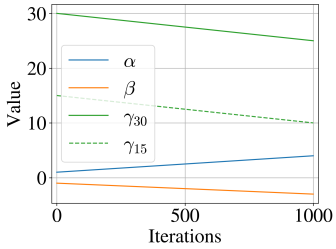
Although the motivation for dynamically changing hyperparameters during the run is valid, DDCFR incurs an overhead of additional compute and time required for feature calculations, policy training, and network inference for real-time computation of discounting weights. In particular, training the RL agent in their work took *24 hours on 200 CPU cores*. Although the training time can be amortized if the trained discounting policy is reused across many games,



(a) Evolution of the weight $(t/(t+1))^\gamma$ applied to the contribution to the average strategy.



(b) Effect of γ on the end exploitability of HS-DCFR at iteration 1,000.



(c) Proposed HSs with a notably large γ for 1,000 iterations.

Figure 1: Motivation and design of Hyperparameter Schedules (HSs).

when the characteristics of the games change over time or when new games with different dynamics are introduced, the pre-trained RL agent might not generalize well to the changes and necessitate retraining. Moreover, when using the DDCFR framework, *multiprocessing* is required to run feature calculations concurrently to avoid increasing wall-clock time. In computationally constrained environments, these additional computational costs are inevitable. Beyond the computational overhead, the RL-based approach also introduces randomness inherent to the training process, so the performance of the learned discounting scheme can only be characterized within a confidence interval.

To leverage the benefits of a dynamic discounting scheme without the complexity and computational overhead of training and running an RL agent, we introduce the concept of *Hyperparameter Schedules (HSs)*—discounting schemes that control how hyperparameters change across iterations—and investigate whether this lightweight, training-free approach can match or even surpass prior SoTA performance.

3.1 Motivation Behind HS$_\gamma$

Although the aforementioned methods achieve strong practical performance, their discounting schemes are not sufficiently aggressive during the early stages of iterations. As a result, they assign substantial and persistent weight to the rough strategies produced in the initial iterations. To quantify this effect, Figure 1a illustrates the evolution of the weight $(t/(t+1))^\gamma$, which determines the contribution of each iteration to the average strategy across different algorithms. As shown, all discounted CFR variants exhibit an inherent transition: early-stage updates contribute less to the average strategy, while later-stage updates have a greater influence (i.e., higher weight). However, there are notable differences in the *rate* of transition. With a fixed $\gamma = 2$, both DCFR and PCFR$^+$ rapidly approach a 0.9 weight within just 19 iterations. DDCFR, which employs a dynamic schedule for γ, reaches the same threshold by 47 iterations—still within only 5% of total training steps. Thus, the applied weight in prior methods converges to near 1 extremely early, meaning that, albeit discounted, the influence of early iterations on the average strategy is still strong.

Our key motivation is the need to delay the convergence of the weight toward 1 in order to *aggressively* suppress the influence of early iterations and *gradually* increase the contribution of later updates as CFR progresses. To counteract the impact of unrefined strategies from early iterations, we design our HSs for γ to start at a very high value (e.g., 30). This forces the discount factor to stay well below 1 for hundreds of iterations, imposing a short memory that effectively suppresses the influence of early updates on the cumulative average strategy. As training progresses, the schedule gradually decreases γ, which increases the discount factor and extends the memory horizon, enabling the algorithm to incorporate the more stable and reliable strategies learned in later iterations. This principled transition from aggressively downweighting early iterations to increasingly trusting late-stage updates is a key driver of the accelerated convergence observed in our experiments (Section 4), and our extensive ablation studies empirically validate this principle.

3.2 Identifying Effective HSs

While our theoretical intuition suggests using a relatively large γ, there are additional intricacies in determining exactly how each schedule should be implemented. In this section, we describe the design choices and implementation details behind our scheduling strategies. We refer to the combination of HS with DCFR as *Hyperparameter Schedule-powered DCFR (HS-DCFR)*, which uses three HSs—one for each of α, β, and γ. Constant schemes originally used in DCFR are special cases of HSs. For example, the empirically best scheme from the DCFR paper can be written as HS$_\alpha = 1.5$, HS$_\beta = 0$, and HS$_\gamma = 2$. We also combine HS with the prior algorithm that is SoTA in non-poker domains, PCFR$^+$. We call that combination *Hyperparameter Schedule-powered PCFR$^+$ (HS-PCFR$^+$)*. As discussed above, PCFR$^+$ uses one fixed HS for γ, namely HS$_\gamma = 2$.

To identify effective HSs for HS-DCFR and HS-PCFR$^+$, we first experimented with large initial values of γ ranging from 10 to 50 and found that HS-DCFR consistently outperformed DCFR across all tested games (see Section 4.1 for game descriptions). In games such as Goofspiel-3, Goofspiel-4 (lim. info.), and Big Leduc poker, performance improves (i.e., achieving lower end exploitability) with increasing γ. However, for games such as Kuhn poker, Battleship-2, Battleship-3, other Goofspiel variants, and the Liar's dice family, performance initially improves and then

degrades as γ becomes too large. Figure 1b illustrates the effect of the initial value of γ in HS on the end exploitability of HS-DCFR at 1,000 iterations. In Kuhn poker, initializing γ at 30 yields the best result, whereas in Goofspiel-4, $\gamma = 15$ is the preferred choice. Notably, for Goofspiel-4 (lim. info.), larger γ values consistently lead to better performance, with $\gamma = 30$ already offering significant improvements.

Therefore, we present two highly performant sets of HSs with a notably large γ starting at 30 or 15, representing two distinct transition rates. As shown in Figure 1a, HS(15) reaches a weight of 0.9 after 136 iterations, while HS(30) reaches a weight of 0.9 after 272 iterations—approximately one-third of the total iterations. For DCFR, we also need to design HSs for α and β. However, our empirical findings indicate that the convergence speed of DCFR toward a Nash equilibrium is significantly more sensitive to the schedule of γ than to those of α and β (see Section 4.2 and the extended version). Therefore, we take inspiration from the parameter ranges used in DDCFR. We denote the two HS-DCFR variants as HS-DCFR(30) that uses (HS_α, HS_β, $\text{HS}_{\gamma 30}$) and HS-DCFR(15) that uses (HS_α, HS_β, $\text{HS}_{\gamma 15}$). Figure 1c illustrates each of the HSs and Equation 4 shows the exact formula, where t is the current iteration and n is the total number of iterations:

$$\begin{aligned} &\text{HS}_\alpha : \alpha = 1 + \frac{3}{n}\,t, && \text{HS}_\beta : \beta = -1 - \frac{2}{n}\,t, \\ &\text{HS}_{\gamma 30} : \gamma_{30} = 30 - \frac{5}{n}\,t, && \text{HS}_{\gamma 15} : \gamma_{15} = 15 - \frac{5}{n}\,t. \end{aligned} \tag{4}$$

For example, to implement HS-DCFR(30), one simply plugs the formulae from Equation 4 into Equations 2 and 3 for α, β, and γ. Additionally, we apply the same two HSs for adjusting γ to PCFR$^+$, yielding HS-PCFR$^+$(30) that uses $\text{HS}_{\gamma 30}$ and HS-PCFR$^+$(15) that uses $\text{HS}_{\gamma 15}$. Given that γ is the only adjustable hyperparameter in PCFR$^+$, HSs for α and β are not used in this setting.

We do not claim that the proposed HSs are optimal for any specific game, as there are infinitely many possible scheduling curves and our schedules are not tuned to individual games. Rather, we present two example schedules to demonstrate that it is possible to design training-free, easy-to-implement, and computationally efficient approaches (requiring only minimal overhead of accessing pre-determined schedule values at runtime) that significantly accelerate CFR convergence across a diverse set of games. We provide the exact formulae in Equation 4 so that readers can reproduce and adopt them for faster game solving, as the required code modifications are minimal (fewer than 15 lines). Others may further build on our schedules, using additional time and computational resources to develop more fine-tuned variants for each game or game class.

3.3 Theoretical Analysis of Convergence Rates

We prove the worst-case convergence rate of HS-DCFR given that the hyperparameters α, β, and γ fall within certain ranges. We also establish the worst-case convergence rate of HS-PCFR$^+$ given that the hyperparameter γ falls within a certain range. Both proofs adopt a common simplification that is used in prior work (Tammelin 2014; Brown and Sandholm 2019a; Xu et al. 2024), namely assuming that both players update their regrets simultaneously on each iteration. Proofs are provided in the extended version.

Theorem 3.1. *Suppose T iterations of HS-DCFR, with simultaneous updates, are played in a two-player zero-sum game, and U is the upper bound of γ across all iterations. If $\alpha \in [1, 5]$, $\beta \in [-5, 0]$, and $\gamma \in [0, U]$, the weighted average strategy profile is a $(U+1)\Delta|\mathcal{I}|\left(\frac{8}{3}\sqrt{|\mathcal{A}|} + \frac{2}{\sqrt{T}}\right)/\sqrt{T}$-Nash equilibrium.*

Theorem 3.2. *Suppose T iterations of HS-PCFR$^+$, with simultaneous updates, are played in a two-player zero-sum game, and U is the upper bound of γ across all iterations. If $\gamma \in [0, U]$, the weighted average strategy profile is a $(U+1)|\mathcal{I}|O(1)/\sqrt{T}$-Nash equilibrium.*

4 Experiments

In this section, we describe our experimental setup, followed by performance comparisons of HS-DCFR and HS-PCFR$^+$ against prior state-of-the-art algorithms.

4.1 Experimental Setup

For evaluation, we use nine extensive-form IIGs and one normal-form IIG that are widely used as benchmarks in computational equilibrium finding. These games include Kuhn poker, Leduc poker, Liar's dice-4, Battleship-3, Goofspiel-4 (with and without limited information), Blotto (a normal-form IIG), Goofspiel-5 (without limited information), *heads-up no-limit Texas hold 'em (HUNL)* endgame-1, and HUNL endgame-3. Additional results on seven more IIGs are provided in the extended version. This diverse set encompasses games of varying complexity, size, and structure, enabling a comprehensive demonstration of the effectiveness of HS-powered algorithms.

Here, we provide a brief overview of the games and their variants. Kuhn poker (Kuhn 1950) is a classic three-card toy poker game with a single betting round. Battleship-x (Farina et al. 2019) is a simplified traditional board game where players secretly position a single ship on their $2 \times x$ grid and alternate taking shots at each other's ship. Goofspiel-x (Ross 1971) is a bidding card game, where each player has x cards and, by placing sealed bids, attempts to score the highest number of points over x rounds. Goofspiel-x (lim. info.) (Lanctot et al. 2009) is the limited-information variant where the players withhold their card information during each turn. Leduc poker (Southey et al. 2005) is an extended version of Kuhn poker with six cards in the deck divided into two suits. There are two betting rounds and a maximum of two raises per round. Big Leduc poker expands Leduc poker to a deck of 24 cards divided into two suits, allowing a maximum of six raises per round. Liar's dice-x (Lisý, Lanctot, and Bowling 2015) is a game where each player rolls an x-sided die, and the players take turns bidding on the outcome. HUNL endgame-x represents an endgame scenario of heads-up no-limit Texas hold 'em. Blotto (Golman and Page 2009) is a resource allocation game where each player allocates finite resources across multiple battlefields.

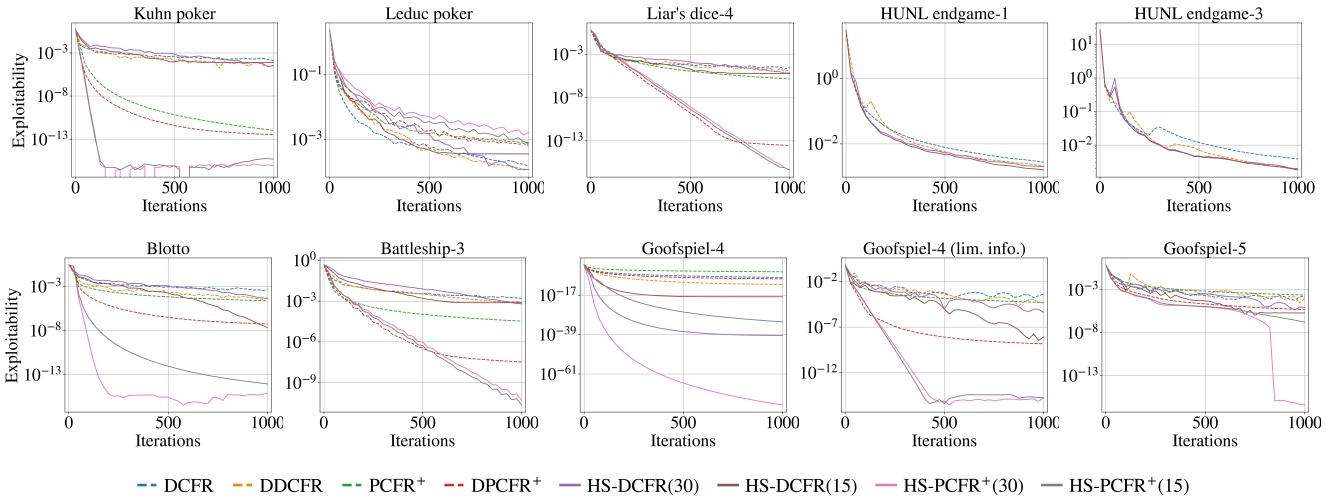


Figure 2: Performance of HS-powered algorithms and prior SoTA algorithms on extensive-form and normal-form games.

We implemented four HS-powered algorithms and to the best of our ability reproduced four baselines using OpenSpiel v1.2 (Lanctot et al. 2019) and PokerRL v0.03 (Steinberger 2019). For HUNL endgames, we plot only the performance of the DCFR variants due to the complexity of reproducing PCFR$^+$ variants in large poker games, as their code is not open-sourced. Notably, as reported in the PCFR$^+$ paper, DCFR outperforms PCFR$^+$ on HUNL endgames. Due to the simplicity of HS, we were able to run the experiments on a single Apple M2 CPU with 16 GB of RAM, with macOS Sequoia v15.5. We set the number of iterations to 1,000, which is sufficient to reach low exploitability and is a typical stopping time used in CFR practice. We used alternating updates in all of the algorithms; that is, in each iteration, both players perform an update one after the other. All algorithms are deterministic; therefore, each algorithm is run once and no confidence intervals are reported.

4.2 Experimental Results

In this section, we present our experimental results for extensive-form games, demonstrating that our proposed algorithms constitute the new SoTA. In Figure 2, we compare four HS-powered algorithms (HS-DCFR(30), HS-DCFR(15), HS-PCFR$^+$(30), and HS-PCFR$^+$(15)) against the prior SoTA algorithms (DCFR, DDCFR, PCFR$^+$, and DPCFR$^+$). These results show that HS-based methods achieve strong performance across a diverse set of games. The new algorithms outperform the prior SoTA by many orders of magnitude on many games and are at least competitive on the remaining ones. To quantify the performance gap, we compute the ratio between the minimum exploitability ($\mathcal{E}$) achieved by DCFR, DDCFR, PCFR$^+$, or DPCFR$^+$ and that achieved by HS-PCFR$^+$(30). We then take the logarithm (base 10) of this ratio, defining the *order of magnitude (OoM)* difference as $\log_{10}(\mathcal{E}_{\text{SoTA}}/\mathcal{E}_{\text{HS-PCFR}^+(30)})$, where $\mathcal{E}_{\text{SoTA}} = \min(\mathcal{E}_{\text{DCFR}}, \mathcal{E}_{\text{DDCFR}}, \mathcal{E}_{\text{PCFR}^+}, \mathcal{E}_{\text{DPCFR}^+})$. Across the ten test games, HS-PCFR$^+$(30) significantly outperforms the prior SoTA by an average of 12.5 *orders of magnitude*. In the following subsections, we discuss the performance of HS-DCFR and HS-PCFR$^+$ in more detail.

HS-DCFR Across the ten benchmark games, HS-DCFR(30) outperforms DDCFR by an average of 3.1 OoM, while HS-DCFR(15) outperforms DDCFR by an average of 1.6 OoM. In HUNL endgames, both HS-DCFR variants consistently achieve strong performance compared to the prior SoTA. These results indicate that simply integrating HS into the original DCFR algorithm yields superior performance compared to DDCFR, despite the latter incurring additional computational costs and complexity. Furthermore, whereas DDCFR employs an RL framework to generate an individual hyperparameter schedule for each game (i.e., game-specific tuning), our approach applies one HS across all games, either (HS_α, HS_β, $\text{HS}_{\gamma 30}$) or (HS_α, HS_β, $\text{HS}_{\gamma 15}$), without any game-specific or game-class-specific tuning.

HS-PCFR$^+$ Across the ten games, HS-PCFR$^+$(30) emerges as the new SoTA, surpassing the prior SoTA by an average of 12.5 OoM. HS-PCFR$^+$(15) also performs strongly, outperforming the prior SoTA by 5.4 OoM on average. Notably, in games like Battleship-3 and Liar's dice-4, the exploitability of the HS-PCFR$^+$ variants decreases at an exponential rate without plateauing by 1,000 iterations. In Goofspiel-5, HS-PCFR$^+$(30) exhibits a substantial drop in exploitability around iteration 800, indicating the discovery of better strategies that traditional CFR variants fail to uncover. Although HS-PCFR$^+$(30) outperforms HS-DCFR(30) by an average of 12.4 OoM across the benchmark games, HS-DCFR(30) is more than an OoM better than HS-PCFR$^+$(30) in Leduc, Big Leduc, and Liar's dice-6. This aligns with the observation in the PCFR$^+$ paper that PCFR$^+$ is less effective than DCFR in poker-like games (except for Kuhn poker, a tiny game). Therefore, we recommend using HS-DCFR(30) for large poker games and HS-PCFR$^+$(30) for the remaining games.

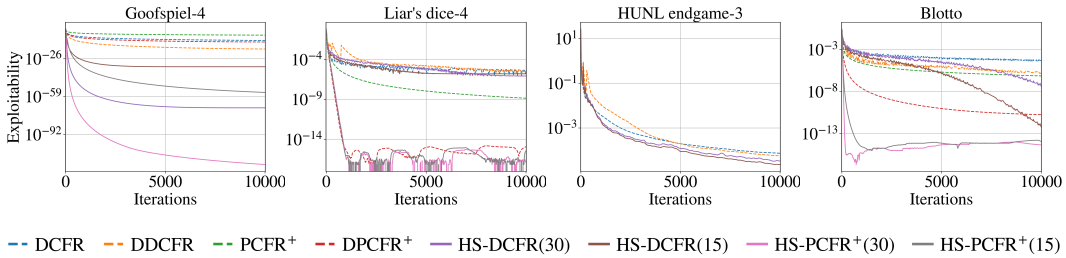


Figure 3: Performance of HS-powered algorithms and prior SoTA algorithms with an extended number of iterations.

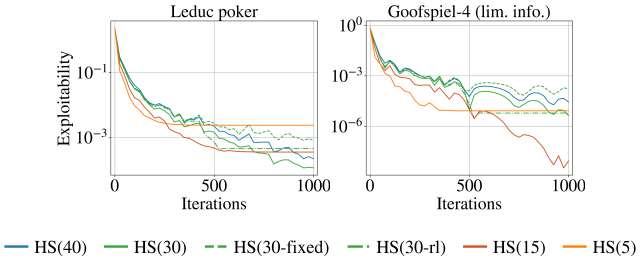


Figure 4: Ablation studies on γ using various HSs.

Normal-Form Game We also test HS-powered algorithms on a common *normal-form* zero-sum benchmark game, General Blotto (Golman and Page 2009). In Blotto, each player allocates finite resources across multiple battlefields, with each battlefield awarded to the player allocating more resources; the final payoff is proportional to the number of battlefields won. We use the setting with two players, three battlefields, and five resources per player. Remarkably, HS-PCFR$^+$(30) surpasses the prior SoTA by 7.9 OoM, establishing the new SoTA on this normal-form zero-sum game by a large margin.

Extended Iterations In Figure 3, we increase the number of iterations to 10,000 when running all algorithms. HS-powered algorithms consistently outperform the prior SoTA by a significant margin. For HUNL endgame-3, we observe a larger performance gain compared to running the algorithms for only 1,000 iterations. In Liar's dice-4, the HS-PCFR$^+$ variants produce a jagged curve, possibly due to instability during updates when regret values are extremely small.

Choice of γ We propose two variants for each of HS-DCFR and HS-PCFR$^+$, using either HS$_{\gamma 30}$ or HS$_{\gamma 15}$. Our experiments show that the two schedules exhibit selective superiority: when HS$_{\gamma 15}$ performs better, the gains are modest, whereas when HS$_{\gamma 30}$ is superior, the improvements are substantial (e.g., in Battleship-3 and Goofspiel-4). On average, HS-DCFR(30) outperforms HS-DCFR(15) by 1.6 OoM, and HS-PCFR$^+$(30) outperforms HS-PCFR$^+$(15) by 7.8 OoM. Therefore, if only one schedule can be used, we recommend HS$_{\gamma 30}$; otherwise, we recommend running both.

Ablation Studies To better understand the strong empirical performance of HS-powered algorithms, we conduct ablation studies to analyze how different HSs of α, β, and γ in HS-powered algorithms influence performance. We consider HS-DCFR that is parameterized by (α, β, γ) and fix α and β according to the HSs proposed in Equation 4. We then examine how the initial value of γ (large or small) or its changing pattern (fixed, linear, or non-linear) across iterations contributes to the effectiveness of HS$_\gamma$. Figure 4 shows the ablation results for γ, and similar studies for α and β are provided in the extended version. Specifically, HS(40) and HS(5) follow the same changing pattern as HS(30) and HS(15) but use initial γ values of 40 and 5, respectively. HS(30-fixed) uses a fixed $\gamma = 30$, while HS(30-rl) follows the γ schedule of DDCFR. We evaluate these variants on both a poker game, Leduc poker, and a non-poker game, Goofspiel-4 (lim. info.). Among the HSs of γ, HS(30) achieves the best performance in Leduc poker while HS(5) performs the worst. The non-linear changing scheme for γ suggested by DDCFR plateaus early and the fixed scheme performs even worse. In Goofspiel-4 (lim. info.), HS(15) yields the best performance, whereas HS(30-fixed) performs the worst, and the non-linear scheme again plateaus early around iteration 500. Thus, combining a well-chosen initial γ with a linear scheme results in strong performance.

5 Conclusions

In this work, we introduced *Hyperparameter Schedules (HSs)*, a simple, training-free framework that dynamically adjusts discounting in CFR variants over time. Motivated by the observation that early strategies are less reliable than those from later iterations, HSs aggressively suppress the influence of early iterations and gradually increase trust in later updates. With fewer than 15 lines of code changes to DCFR and PCFR$^+$, our resulting algorithms, HS-DCFR and HS-PCFR$^+$, establish new SoTA performance in both extensive-form and normal-form settings, often achieving orders of magnitude improvements in convergence rate. We acknowledge that game-specific or game-class-specific tuning could further enhance the performance of HSs. Moreover, in games where the relative advantage of HSs diminishes as the game size increases, such tuning may help counteract this diminishing benefit. Nevertheless, the proposed HSs serve as a strong starting point for future research, as they already achieve SoTA performance across a wide range of games even without game-specific tuning.

Acknowledgments

This material is based on work supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, National Science Foundation grant RI-2312342, ARO award W911NF2210266, and NIH award A240108S001. Stephen McAleer was supported by a CRA Computing Innovation Fellow postdoctoral fellowship.

References

Blackwell, D. 1956. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 1–8.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science*, 145–149.

Brown, N.; and Sandholm, T. 2015. Regret-based pruning in extensive-form games. *Conference on Neural Information Processing Systems (NeurIPS)*.

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

Brown, N.; and Sandholm, T. 2019a. Solving imperfect-information games via discounted regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 1829–1836.

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

Cesa-Bianchi, N.; and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge University Press.

Farina, G.; Kroer, C.; and Sandholm, T. 2021. Faster game solving via predictive Blackwell approachability: Connecting regret matching and mirror descent. In *AAAI Conference on Artificial Intelligence (AAAI)*, 5363–5371.

Farina, G.; Ling, C. K.; Fang, F.; and Sandholm, T. 2019. Correlation in extensive-form games: Saddle-point formulation and benchmarks. *Conference on Neural Information Processing Systems (NeurIPS)*.

Golman, R.; and Page, S. E. 2009. General Blotto: games of allocative strategic mismatch. *Public Choice*, 279–299.

Hart, S.; and Mas-Colell, A. 2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 1127–1150.

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

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. *arXiv preprint arXiv:1908.09453*.

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. *Conference on Neural Information Processing Systems (NeurIPS)*.

Lisý, V.; Lanctot, M.; and Bowling, M. H. 2015. Online Monte Carlo Counterfactual Regret Minimization for Search in Imperfect Information Games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 27–36.

McAleer, S. M.; Farina, G.; Lanctot, M.; and Sandholm, T. 2023. ESCHER: Eschewing Importance Sampling in Games by Computing a History Value Function to Estimate Regret. In *International Conference on Learning Representations (ICLR)*.

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

Nash, J. 1950. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 48–49.

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

Ross, S. M. 1971. Goofspiel—the game of pure strategy. *Journal of Applied Probability*, 621–625.

Southey, F.; Bowling, M. P.; Larson, B.; Piccione, C.; Burch, N.; Billings, D.; and Rayner, C. 2005. Bayes' bluff: Opponent modelling in poker. In *Annual Conference on Uncertainty in Artificial Intelligence (UAI)*.

Steinberger, E. 2019. PokerRL. https://github.com/TinkeringCode/PokerRL.

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

Waugh, K.; Schnizlein, D.; Bowling, M. H.; and Szafron, D. 2009. Abstraction pathologies in extensive games. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 781–8.

Xu, H.; Li, K.; Fu, H.; Fu, Q.; Xing, J.; and Cheng, J. 2024. Dynamic discounted counterfactual regret minimization. In *International Conference on Learning Representations (ICLR)*.

Zarick, R.; Pellegrino, B.; Brown, N.; and Banister, C. 2020. Unlocking the potential of deep counterfactual value networks. *arXiv preprint arXiv:2007.10442*.

Zhang, H.; Lerer, A.; and Brown, N. 2022. Equilibrium finding in normal-form games via greedy regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 9484–9492.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. *Conference on Neural Information Processing Systems (NeurIPS)*.