Regret-Minimizing Double Oracle for Extensive-Form Games

Xiaohang Tang [1] **Le Cong Dinh** [2] **Stephen Marcus McAleer** [3] **Yaodong Yang** [4]

Abstract

By incorporating regret minimization, double oracle methods have demonstrated rapid convergence to Nash Equilibrium (NE) in normal-form games and extensive-form games, through algorithms such as online double oracle (ODO) and extensive-form double oracle (XDO), respectively. In this study, we further examine the theoretical convergence rate and sample complexity of such regret minimization-based double oracle methods, utilizing a unified framework called Regret-Minimizing Double Oracle. Based on this framework, we extend ODO to extensive-form games and determine its sample complexity. Moreover, we demonstrate that the sample complexity of XDO can be exponential in the number of information sets $|S|$, owing to the exponentially decaying stopping threshold of restricted games. To solve this problem, we propose the Periodic Double Oracle (PDO) method, which has the lowest sample complexity among regret minimization-based double oracle methods, being only polynomial in $|S|$. Empirical evaluations on multiple poker and board games show that PDO achieves significantly faster convergence than previous double oracle algorithms and reaches a competitive level with state-of-the-art regret minimization methods.

1. Introduction

Extensive-form games (EFGs) have been extensively employed in modeling sequential decision-making problems, including auctions, security games, and poker. However, solving such games remains challenging in real-world applications due to various complexities, including the game's large size and imperfect information. While linear programming (LP) (Von Neumann & Morgenstern, 1947) and sequence-form LP (Koller & Megiddo, 1992) can be used to compute the Nash equilibrium (NE) of EFGs (Nash Jr, 1950), direct calculation of the exact NE via LP can become infeasible for large real-world games due to memory constraints and the high cost of matrix inversion. Therefore, more efficient methods are required to solve EFGs for real-world applications.

The Double Oracle (DO) (McMahan et al., 2003) algorithm family has been developed to address the complexity of solving large Extensive-Form Games (EFGs) by solving a sequence of restricted games, where players can only select actions from a subset of pure strategies in the original game. These restricted games are typically much smaller than the original EFG, especially when the support of the Nash equilibrium (NE) is small (Wilson, 1972; Koller & Megiddo, 1996; Bosansky et al., 2014). For instance, in symmetric normal-form games with random entries, the NE's support size is only half of the game size (Jonasson et al., 2004). By constructing a solution to the large game using only a small subset of pure strategies, the DO algorithm can effectively solve large EFGs. The DO algorithms can also be combined with deep reinforcement learning (Lanctot et al., 2017; Muller et al., 2019; McAleer et al., 2022b;a) to achieve state-of-the-art performance on large games like Stratego (McAleer et al., 2020) and Starcraft (Vinyals et al., 2019).

Recent advances in game theory have led to the development of methods that integrate the Double Oracle (DO) algorithm with regret minimization techniques. For example, in large normal-form games, the Online Double Oracle (ODO) algorithm combines the DO approach with the Multiplicative Weights Update (MWU) (Freund & Schapire, 1999) regret minimizer for restricted games (Dinh et al., 2022). In extensive-form games, the Counterfactual Regret Minimization (CFR) algorithm has been successfully used to achieve superhuman performance in Texas Hold'em Poker and other games (Zinkevich et al., 2007; Brown et al., 2019; Lanctot et al., 2009; Farina et al., 2020; McAleer et al., 2023). To further improve the performance of DO in this setting, the Extensive-Form DO (XDO) algorithm applies CFR to the DO framework and can support games with high-dimensional continuous action spaces (McAleer et al., 2021). However, theoretical questions regarding the convergence speed, expected iterations, and sample com-

[1]Department of Statistical Science, University College London [2]University of Southampton [3]Carnegie Mellon University [4]Institute for AI, Peking University. Correspondence to: Yaodong Yang <yaodong.yang@pku.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

plexity of DO methods in EFGs have yet to be thoroughly investigated and remain an open area of research (Bosansky et al., 2014).

In this study, a general theoretical framework is presented to investigate algorithms that combine DO and regret minimization, providing their expected number of iterations and sample complexity to reach ϵ-NE. This is to our knowledge the first work analyzing the theoretical convergence rate of DO in EFGs. The paper presents two applications of the framework. In the first example, we extend ODO to solve EFGs and determine the corresponding sample complexity. In the second example, we prove that the stopping condition for restricted games in the Extensive-Form DO (XDO) algorithm may lead to a worst-case convergence in a number of iterations exponential in the number of information sets $|S|$. To address this issue, we propose Periodic Double Oracle (PDO), which only has a polynomial sample complexity in $|S|$, lower than that of both ODO and XDO. Empirical assessments on typical poker and board games demonstrate that PDO achieves much faster convergence compared to XDO and ODO, reaching a level competitive with state-of-the-art regret minimization methods.

2. Preliminaries

2.1. Two-Player Zero-Sum Extensive-Form Games

In this paper, we focus on Two-Player Zero-Sum Extensive-Form Games (**EFGs**) with perfect recall. Our notation is based on that of Brown (2020). EFGs are represented by a game tree where nodes correspond to players $i \in \mathcal{P} = \{1, 2\}$. Imperfect Information EFGs employ **chance player** c to model the stochastic events like card dealing in Poker. **History** (h) is a sequence of actions the players took and events uniquely attached to a node on the game tree. $A(h)$ is the set of available actions at h, and $P(h)$ denotes the player who needs to make a decision at h. Terminal histories, which are all in the set Z, are attached to nodes where the game's payoff can be determined. The payoff at the terminal history $z \in Z$ is treated as value at z, denoted as $v_i(z)$. The range of the payoff is represented by .

Information Set (Infoset/Infostate s_i**)** of each player $i \in \mathcal{P}$ is the set of indistinguishable histories from player i's perspective. The set S_i contains infosets where player i must make decisions, and S represents the union of information sets for all players: $S = \cup_{i2P} S_i$. The set $A(s_i)$ includes all available actions of player i at s_i. The **Strategy** of player i is represented by π_i, with $\pi_i(s_i, a)$ representing the probability of player i's taking action $a \in A(s_i)$ at the information set s_i. Joint strategy $\pi = (\pi_1, \pi_2)$. **Reaching probability** $x\ (h)$ is the probability to reach h when players use joint strategy π. Specifically, $x\ (h) = \prod_{i2P \cup fcg} x_i\ (h)$, where $x_i\ (h) = \prod_{h^0\ a\ h} \pi_{P(h^0)}(h^0, a)$ is player i's contribution. Given joint strategy $\pi = (\pi_1, \pi_2)$, we can define the **expected value** of history h of player i, denoted by $v_i(h)$. If reaching probability $x\ (h) = 0$, then $v_i\ (h) = 0$; Otherwise, we have

$$v_i\ (h) = \sum_{z2Z} \frac{x\ (z)}{x\ (h)} v_i(z),\ x\ (h) > 0, \tag{1}$$

and thus the value function is *linear* with respect to strategy π:

$$v_i(\pi_i, \pi\ _i) = \sum_{z2Z} v_i\ (z) x\ (z). \tag{2}$$

Then **best response** $\mathsf{BR}_i(\pi\ _i)$ is $\arg\max\ _i v_i(\pi_i, \pi\ _i)$. ϵ-**Nash Equilibrium (NE)** strategy π satisfies that $\forall i \in \mathcal{P}$, $\min\ _i v_i(\pi_i, \pi\ _i) + \epsilon \geq v_i(\pi\ ) \geq \max\ _i v_i(\pi_i, \pi\ _i) - \epsilon$. In particular, exact NE is ϵ-NE when $\epsilon = 0$. **Exploitability** $e(\pi) = \sum_{i2P} v_i(\mathsf{BR}(\pi\ _i), \pi_i) - v_i(\pi)$. **Support Size** of NE $(\pi\ )$ denotes the number of actions that have positive probabilities at infoset s in NE $\pi\ $, denoted by supp (s) (Schmid et al., 2014).

2.2. Regret Minimization Algorithms

Regret minimization methods approximate NE in EFGs if the algorithm has a sublinear regret upper bound or an average regret converging to zero as iteration T goes to infinity. In this context, we introduce different regret minimization algorithms, and we will demonstrate how these algorithms ensure convergence to NE.

Definition 2.1 (Regret). Given $\{\pi^t |\ t = 1, \cdots, T\}$ is a sequence of strategies delivered by an algorithm, the regret of this algorithm is defined as:

$$R_i^T = \sum_{t=1}^{T} \max v_i(\pi, \pi^t\ _i) - v_i(\pi_i^t, \pi^t\ _i), \tag{3}$$

and *average* regret $R_i^T = R_i^T / T$.

Vanilla Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2007) is a regret minimization algorithm that aims to minimize counterfactual regret at each infoset by traversing the full game tree depth-firstly. It achieves this by calculating player i's expected values $v_i(\cdot)$ based on equation 1 and computing instantaneous regrets at iteration $t \leq T$ of taking action a in infoset s following $r_i^t(s, a) = \sum_{h2s} x\ _i^t(h)[v_i\ ^t(h \cdot a) - v_i\ ^t(h)]$. The counterfactual regrets of CFR can be computed by uniform average of r_i over all iterations: $R_i^T(s, a) = \sum_{t=1}^{T} r_i^t(s, a)/T$. In two-player games, if both players apply regret matching (Zinkevich et al., 2007) to strategy updates, the regret of CFR is bounded by $|S_i| \sqrt{|A_i|} \cdot T^{1=2}$.

Discounted regret minimization methods aim to minimize a weighted-average regret in which the strategy in each

iteration is discounted:

$$R_i^T = \sum_{t=1}^{T} w_t \left[\max_{\pi_i'} v_i(\pi_i'; \sigma_{-i}^t) - v_i(\sigma_i^t; \sigma_{-i}^t) \right] = \sum_{t=1}^{T} w_t r_i^t. \tag{4}$$

The discounted CFR (DCFR) (Brown & Sandholm, 2019) is a regret minimization framework that belongs to this family and is based on the counterfactual regret minimization algorithm. DCFR employs weighted-average counterfactual regrets and weighted-average strategy to achieve faster convergence. It has an upper bound on weighted-average regret, where w_t satisfy $\lim_{T\to\infty} \frac{\sum_{t=1}^{T} w_t}{T} = 1$. The weighted-average regret's upper bound is $|S_i|(|\overline{A_i}| + \frac{1}{T})\sqrt{T}$. DCFR can be generalized to other CFR variants such as vanilla CFR, CFR+, and Linear CFR with appropriate hyperparameters (Tammelin, 2014; Brown & Sandholm, 2019; Brown, 2020).

For the purpose of facilitating analysis, we adopt a uniform notation for the upper bound of regret throughout the remaining sections of this paper. Specifically, we use $O(|S_i|\sqrt{|A_i|}T^{1/2})$ as the average regret bound of CFR algorithms including vanilla CFR and DCFR.

2.2.1. Regret Minimization to Nash Equilibrium

A widely accepted folk lemma suggests that if both players in a two-player zero-sum game adopt an algorithm with a sublinear regret bound, then the average strategies of both players will converge to a Nash Equilibrium (Cesa-Bianchi & Lugosi, 2006; Blum & Monsour, 2007). Here we prove the discounted regret version with the same idea.

Lemma 2.2. Given the weighted-average regret of an algorithm $O(|S_i|\sqrt{|A_i|}/\sqrt{T})$, the weighted-average strategy of this algorithm is an $O(|S_i|\sqrt{|A_i|}/\sqrt{T})$-NE.

The proof is in Appendix B.2. Therefore, if the weighted-average regret can converge to zero, the resulting weighted-average strategy is a reliable approximation that converges to NE.

2.3. Double Oracle Methods

Double Oracle (DO) (McMahan et al., 2003) is a technique initially developed for solving Normal-form Games (NFGs) which maintains a population of pure strategies for both players, denoted as Π_t at time t, and creates a restricted game by considering only the actions in Π_t. Then the restricted game's NE is obtained using linear programming, and the best response to this NE is added to the population. This process repeats until the population no longer changes. DO's advantage is that it solves large games by only solving some small restricted games, since the restricted game usually stops growing when it is still small, particularly when the support of the NE is small (i.e., non-zero probabilities of NE strategy's actions are few) (Wilson, 1972; Koller & Megiddo, 1996). Table 2 in Appendix C includes the support size of NE in some common games. However, traditional algorithms such as linear programming can still be intractable in large restricted games. To address this issue, Online Double Oracle (ODO) (Dinh et al., 2022) uses regret minimization for strategy updates in the restricted games, leading to better empirical performance than DO in large normal-form games.

Algorithm 1 XDO (McAleer et al., 2021)

Input: initial threshold ϵ_0, time window index $j = 0$, uniform random strategy σ^0.
$\Pi_1 = [_{i\in\{1,2\}} \text{BR}_i(\sigma_{-i}^0)$.
Construct restricted game G_1 with Π_1.
for $t = 1, \ldots, T-1$ **do**
  Run one iteration of CFR in G_t.
  if Exploitability in G_t, $e(\sigma^t) \le \epsilon_0/2^j$ **then**
    $\Pi_{t+1} = \Pi_t [\text{BR}(\sigma_{-i}^t)$ for $i \in \{1, 2\}$.
    $j = j + 1$.
    Reset σ^{t+1}.
    Construct restricted game G_{t+1} with Π_{t+1}.
  end if
end for

The conversion of extensive-form games (EFGs) into normal-form games and solving them with normal-form Double Oracle (DO) is theoretically feasible. However, representing EFGs in normal-form will result in an exponential increase in the number of required iterations for convergence (McAleer et al., 2021). Therefore, a specific DO algorithm for EFGs is necessary. The first DO method for EFGs, Sequence-form DO (SDO), was proposed by Bosansky et al. (2014). This approach uses the Sequence-form LP to compute the exact NE of the restricted game. Moreover, SDO introduces an efficient technique for expanding the restricted game in extensive-form algorithms, where each iteration involves adding sequences instead of one pure strategy to the population. To address the challenge of solving large restricted games, McAleer et al. (2021) extends SDO to Extensive-Form DO (XDO), which employs CFR to approximate the restricted game's NE and reinforcement learning methods to approximate the best response. Since XDO adds actions at every information set, it expands the restricted game much faster than SDO. The formal process of XDO is in Algorithm 1, which is rearranged to count one iteration as the completion of one strategy update in the restricted game, followed by computing a BR if required. For more details on Algorithm 1, please refer to Appendix A.2.

To demonstrate the efficacy of DO for EFGs, we present a basic example of XDO (SDO) in a two-player zero-sum

EFG, as depicted in Figure 4. DO methods can compute the NE in EFGs with a small support NE without solving the original game. XDO has been empirically shown to converge rapidly in small support games (McAleer et al., 2021), but it lacks a theoretical analysis of the convergence rate for achieving an approximate NE. In the following section, we introduce a general framework that can generalize to both XDO and extensive-form ODO, and provide a theoretical analysis of their convergence rates. Based on the analysis, we propose a more sample-ef cient DO algorithm for EFGs.

3. Regret-Minimizing Double Oracle

In this section, we propose Regret-Minimizing Double Oracle (RMDO), a novel generic Double Oracle framework combined with regret minimization to approximate the Nash Equilibrium of EFGs. To the best of our knowledge, this is the rst study analyzing the convergence rate and sample complexity of regret-minimization based Double Oracle for EFGs.

RMDO consists of the same elements as the previous DO methods. Restricted game is constructed by considering only a subset of all pure strategies. Population Π containing the available pure strategies in the restricted game. Time window T_j, de ned as a partition of the set of all iterations where the populations are the same: $\forall t_0, t_1 \in T_j,\ \Pi_{t_0} = \Pi_{t_1}$, plays a crucial role in RMDO and contributes to making it a generic framework. The number of time windows, denoted by k, corresponds to the number of restricted games from iteration $t = 0$ to T. However, in contrast to existing DO methods, RMDO has the ability to expand the restricted game at any time.

Prior to presenting the formal process, we highlight two key new components of RMDO. The rst component is the frequency function $m(\cdot)$ used in the computation of Best Response, denoted as $\sigma(\cdot)$, which is de ned as a mapping from the set of time window indices $\mathbb{N} \setminus [0, k-1]$ to $\mathbb{N}^+$. The function $m(j)$ represents the frequency of computing Best Response in the jth time window. Since the process of DO based on regret minimization is exactly to take turns to do regret minimization and compute the best response, balancing the regret minimization and Best Response computation is critical to achieve a rapid convergence. The second component is weighted-average scheme. To accelerate the convergence in the restricted game, we incorporate within-window weights w_t, where $t \in T_j$, allowing us to utilize the discounted regret minimizer. The within-window weights are exactly the weights in the discounted regret minimizer, satisfying that their sum in the current window T_j equals one. For instance, vanilla CFR employs uniform weights $1/|T_j|$ in window T_j. CFR+ and linear CFR use linearly increasing weights. Speci cally, within the jth window T_j:

$$w_t = 2(t - \sum_{m=1}^{j-1} |T_m| + 1)/|T_j|(|T_j| + 1). \quad (5)$$

Thus, the weighted average strategy in the window T_j is:

$$\tilde{\sigma}_i^t = \sum_{t' \in T_j} \sigma_i^{t'} w_{t'}. \quad (6)$$

Presented in Algorithm 2, the formal RMDO procedure is as follows. At each iteration t, assuming the current time window is j, the restricted game G_t is constructed by restricting the pure strategies in the population Π for players. Within G_t, regret minimization is conducted by traversing the game tree, computing the regret of each infoset (node), and updating the strategy using any Counterfactual Regret Minimization (CFR) algorithm. At the outset of the procedure, when $t = 0$, the construction of the restricted game and the strategy update are bypassed since Π_0 is empty. The expected value at $t = 0$ is computed based on the joint strategy σ^0 following a uniformly random policy. As the procedure progresses, when $t > 0$ and the current time window is T_j, the joint average strategy of current window $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2)$ is expanded to the original game every $m(j)$ iteration by setting the probabilities of actions not in the population to zero. Then the original game best response (BR), considering all actions in the original game, is computed against the expanded current-window average strategy, which is $a_i^t = \arg\max_{a_i \in A_i} v_i(a_i, \tilde{\sigma}_i)$, for both players. a_i^t for $i = 1, 2$ are both merged to the population Π_{t+1}. Finally, if the population changes ($\Pi_{t+1} \neq \Pi_t$), a new time window is initiated, and σ_i^{t+1} is reset to a uniform random strategy.

Algorithm 2 Regret-Minimizing Double Oracle

Input: hyperparameter m, window index $j = 0$, uniform random strategy σ^0.
Set population $\Pi_1 = BR_i(\sigma^0)$ for $i \in \{1, 2\}$.
Construct restricted game G_1 with Π_1.
for $t = 1, \ldots, T-1$ **do**
  Run one iteration of CFR in G_t.
  if $t \bmod m(j) = 0$ **then**
    Compute $\tilde{\sigma}_i^t$ with equation (6).
    $\Pi_{t+1} = \Pi_t \cup [BR_i(\tilde{\sigma}_i^t)]$ for $i \in \{1, 2\}$.
    if $\Pi_{t+1} \neq \Pi_t$ **then**
      Start new window: $j = j + 1$.
      Reset strategy σ^{t+1}.
      Construct restricted game G_{t+1} with Π_{t+1}.
    end if
  end if
end for

Then we investigate the convergence guarantee of RMDO. Regret minimization algorithms can converge to NE by

iteratively updating strategy in a static game. But in RMDO, a regret minimizer is employed in the restricted game, which expands over time. Thus if the restricted game stops expanding at some finite iteration, the convergence of RMDO is guaranteed. The following lemma proves that the number of restricted games is finite, which guarantees RMDO's convergence.

Lemma 3.1. In an extensive-form game, where represents the set of Nash Equilibrium (NE) strategies of this game. When running RMDO, given that is the number of restricted games, we have $\min_{\sigma^* \in \Sigma^*} \max_{s \in S} \text{supp}(\sigma^*(s)) < k \sum_i |S_i|$. (Refer to Appendix B.1 for proof).

RMDO will converge by doing regret minimization in the final restricted game, after times of restricted game expanding. Additionally, according to Lemma 3.1, where is bounded, the subsequent assumption can be made in hindsight, without any loss of generality, for the remainder of this paper.

Assumption 3.2. Given an extensive-form game solving by Regret-Minimizing Double Oracle, there are restricted games even when $T \to \infty$.

While the value of is unknown during training, it is possible to partition the iterations into a limited number of time windows in hindsight to investigate the convergence rate and sample complexity of RMDO. In the following section, we will examine two types of strategies that are generated by RMDO.

3.1. Overall Average Strategy

The convergence rate of the average strategy can be determined by the regret upper bound of the regret minimization algorithm. Thus we first investigate the overall average strategy by taking the average over iteration $t = 0$ to T.

In addition to the within-window weights, global weights are assigned to $_t$ when computing the overall average strategy. Specifically, for all $t \in T_j$ within a time window T_j, the weight W_t is defined as $|T_j| w_t / T$. It can be easily shown that $\sum_{t=1}^{T} W_t = 1$. The overall average strategy for player i is then obtained as:

$$\bar{\sigma}_i^T = \sum_{t=1}^{T} \sigma_i^t W_t = \sum_{t=1}^{T} \sigma_i^t \frac{|T_j| w_t}{T}. \quad (7)$$

Define weighted-average regret of RMDO as

$$R_i^T = \max_{\sigma_i^0} \sum_{t=1}^{T} \left(v_i(\sigma_i^0, \sigma_{-i}^t) - v_i(\sigma^t)\right) W_t. \quad (8)$$

We then prove that the regret has the following upper bound:

Theorem 3.3. In RMDO, suppose the regret minimizer has $O(|S_i| \sqrt{|A_i| T})$ regret, the weighted-average regret bound of RMDO:

$$O\left(\sum_{j=0}^{k-2} \frac{|T_j|}{T} [m(j) - 1] + \sum_{j=0}^{k-1} \frac{\sqrt{k} |S_i| |T_j|}{T \sqrt{|T_j| - m(j) + 1}}\right), \quad (9)$$

converges to 0 if $m(j)$ is sublinear in T.

The proof is in Appendix B.3. If the number of restricted games $m(j)$ is sublinear in the total number of iterations T, then the Regret-Minimizing Double Oracle (RMDO) algorithm is an anytime algorithm that converges to Nash Equilibrium (NE) with its overall average strategy. Moreover, with the help of Lemma 2.2, one can easily obtain the expected number of iterations required for the algorithm to converge to -NE.

3.2. Last-Window Average Strategy

The final average strategy obtained from regret minimization in the last time window is an -NE, requiring at least $O(|A_{i,k}| |S_{i,k}|^2 / \varepsilon^2)$ iterations to reach, where $A_{i,k}$ and $S_{i,k}$ denote the action and information set space in the last time window T_k of player i. The regret bound of the regret minimizer is $O(|S_{i,k}| \sqrt{|A_{i,k}| |T|})$. During the growth of the population, we observe that in each time window, the regret minimizer at each iteration except the last one will not reach -NE. Otherwise, in a non-last time window, the global best response is already in the population. Thus, the average strategy in this window at this iteration will already reach -NE, which is contradictory. Utilizing this idea, we can bound the number of iterations required for each time window and provide a sample complexity to reach NE.

Theorem 3.4. The last-window average strategy of RMDO needs the following number of iterations to reach NE:

$$O\left(k|A||S|^2/\varepsilon^2 \left(k + \sum_j m(j)\right)\right). \quad (10)$$

The proof is in Appendix B.4. Utilizing this result, we can estimate the sample complexity required for RMDO to achieve -NE. It is noteworthy that as the exact regret minimizer will traverse the entire game tree, the sample complexity is at most $O(|S|)$. Hence, the sample complexity for the regret minimization part is given as $O\left(k|A||S|^3/\varepsilon^2 \left(k|S| + |S| \sum_j m(j)\right)\right)$.

In addition, we also need to investigate the sample complexity involved in computing the best response (BR). As BR computation also requires full tree traversal, we only need to consider the number of times RMDO computes BR during training. Since it is reliant on the selection of frequency function $m(j)$, we will examine the overall sample complexity in the following section where we introduce RMDO

Algorithm 3 Extensive-Form Online Double Oracle

Initial empty population Π_0, time window T_0
for $t = 0, \ldots, T-1$ do
  Construct restricted game G_t with Π_t.
  Update strategy σ^t in G_t.
  for $i = 1, 2$ do
    Get BR: $a_i^t = \arg\max_{\pi_i \in \Pi_i} \sum_{t' \in T_j} v_i(\pi_i, \sigma_i^{t'})$.
    Update population $\Pi_{t+1} = \Pi_t \cup \{a_i^t\}$.
  end for
  if $\Pi_t = \Pi_{t+1}$ then
    Start new window T_{j+1} and Reset strategy σ_i^{t+1}.
  end if
end for

with various schemes of frequency functions and demonstrate how it generalizes to existing methods. Following this analysis, we propose a more sample-efficient algorithm in comparison to existing methods.

4. Efficient Schemes of Frequency Function

The complexity of approximating NE using RMDO is influenced by the choice of frequency function $m(j)$ for best response computation, as demonstrated in the previous section through theoretical analysis. For analysis on existing methods, we present various RMDO instantiations with distinct frequency schemes in this section.

4.1. Online Double Oracle for Extensive-Form Games

We propose an extension of the Online Double Oracle (ODO) algorithm, known as the Extensive-Form Online Double Oracle (XODO), which combines the Sequence-form DO framework with Counterfactual Regret Minimization (CFR) to solve extensive-form games. The algorithm is described in detail in Algorithm 3, where the construction and update of the restricted game and strategy are performed in a similar manner to the DO framework used in ODO. However, in each iteration, XODO expands the restricted game by computing the best response against the average strategy in the current window. As XODO computes the best response in each iteration after regret minimization, it is equivalent to the RMDO algorithm with $m(j) = 1$. Based on Theorem 3.3, XODO has a regret bound.

Corollary 4.1. In XODO, given the regret minimizer with $O(|S_i| \sqrt{|A|T})$ regret upper bound, the weighted-average regret bound of XODO will be:

$$O\left(\frac{|S_i| k}{\sqrt{T}}\right). \tag{11}$$

Proof. Plug $m(j) = 1$ into equation 9, the upper bound will be $O(|S_i| \sqrt{k} \sum_j \sqrt{|T_j|}/T)$. According to Cauchy-Schwartz inequality, $\sum_j \sqrt{|T_j|} \le \sqrt{k \sum_j |T_j|} = \sqrt{kT}$, then the upper bound becomes $O(|S_i| k/\sqrt{T})$. □

Then according to lemma 3.1, $\forall j$, $|S_j|$, XODO has a sublinear regret upper bound. According to the regret to strategy conversion in Lemma 2.2, the overall average strategy of XODO requires $O(|S|^2 k^2/\epsilon^2)$ iterations to reach ϵ-NE. We can further derive the sample complexity:

Proposition 4.2. Since XODO compute BR in each iteration, the sample complexity to reach ϵ-NE is $O(2|S|^3 k^2/\epsilon^2)$. (Proof in Appendix B.5).

4.2. Extensive-Form Double Oracle

The Extensive-form Double Oracle (XDO) algorithm is initialized with a given threshold ϵ_0, which is divided by two each time the local exploitability of the regret minimizer meets the threshold. The local exploitability is the exploitability in the restricted game. In time window T_j, the algorithm performs regret minimization for more than $4^j |S_{i,j}|^2 |A_{i,j}|/\epsilon_0^2$ iterations before computing the best response. Here $A_{i,j}$ and $S_{i,j}$ denote the action space and infoset space in the j-th time window of player i. Finally the average strategy in the last window is outputted when the convergence condition is met. If XDO converges, the last-window average strategy is $\epsilon_0/2^k$-NE. To investigate the complexity of reaching ϵ-NE, it is assumed without loss of generality that $\epsilon_0 = 2^k \epsilon$.

Thus, RMDO can generalize to XDO with $m(j) = 4^j |S_{i,j}|^2 |A_{i,j}|/\epsilon_0^2$ and the last-window average strategy. Based on Theorem 3.4, we can determine the expected iterations and sample complexity for XDO to reach ϵ-NE.

Corollary 4.3. XDO needs at least $O(k|A||S|^2/\epsilon^2 + |A||S|^2 4^k/\epsilon_0^2 \cdot k)$ iterations to reach ϵ-NE.

Proposition 4.4. Since XDO computes BR only at the end of the time window before convergence, the sample complexity to reach ϵ-NE is at least $O(k|A||S|^3/\epsilon^2 + |A||S|^3 4^k/\epsilon_0^2)$.

Corollary 4.3 is a specific instance of Theorem 3.4 with an appropriate choice of $m(j)$, and its proof is provided in Appendix B.6. The sample complexity of XDO is analyzed in Proposition 4.4, and its proof can be found in Appendix B.7. Theorem 3.1 states that $\forall j \le |S|$; thus, theoretically, the restricted game stopping condition of XDO decays exponentially, implying that in the worst-case scenario, when $k = |S|$, XDO has an exponential sample complexity in the number of infosests. Therefore, XDO suffers from a large theoretical sample complexity. Empirically, the values of k when executing XDO on common poker games leads to a large sample complexity (Appendix A).

Figure 1. Performance of XODO and PDO with periodicity function $m(j) = 1; 10; 50; 100$ on Leduc Poker, Dummy Leduc Poker and Oshi Zumo.

Figure 2. Performance of PDO with periodicity 50 and XDO on Sequential Blotto Games and Kuhn Poker with initial pot 40 for each player. In Sequential Blotto, XDO is still significantly more exploitable than PDO even after visiting more than 10^6 infosets.

4.3. Periodic Double Oracle

The exponentially growing frequency function $m(j)$ of XDO leads to an exponential increase in sample complexity with respect to k. On the other hand, XODO's inflexibility arises from the fact that it performs best response computation in each iteration, neglecting the balance between regret minimization and best response computation. In order to mitigate the large increase in sample complexity caused by a large value of k, and to balance the two computations, we propose the Periodic Double Oracle (PDO) algorithm. PDO computes the best response at a fixed time interval in each window and outputs the average strategy in the last window.

By setting $m(j)$ to a constant c, PDO can be viewed as an instantiation of RMDO. We can derive the expected number of iterations required for PDO to reach an approximate NE by utilizing Theorem 3.4. The algorithm is presented in Appendix D.

Corollary 4.5. PDO needs $O(k|A||S|^2/\varepsilon^2 + (c-1)k)$ iterations to reach ε-NE.

Proposition 4.6. Since PDO computes BR every c iterations, the sample complexity to reach ε-NE is $O(k|A||S|^3/\varepsilon^2 + ck|S| + k|A||S|^3/c\varepsilon^2 - k|S|/c)$.

Proposition 4.6 has been demonstrated in Appendix B.8. The periodicity $m(j) = c$ in PDO reduces the impact of the dominating term $|S|^3$ in sample complexity, compared to that of XODO. Additionally, compared to XDO, PDO eliminates the term exponential in k from the sample complexity. While XDO may have an exponential sample complexity in $|S|$ in the worst case scenario, PDO only has polynomial complexity in $|S|$. Hence, theoretically, PDO is more sample-efficient than existing algorithms (refer to Table 3 for a summary of sample complexities).

Given that these complexities are computed in the worst-case scenario, and are computed to ensure that the algorithm reaches at most ε-NE, we cannot determine the value of c by merely solving for extreme values of sample complexity. Instead, we consider it as a hyperparameter and analyze the empirical performance of PDO with different c in the next section.

5. Experiments

All experiments and algorithm implementations are based on OpenSpiel (Lanctot et al., 2019). Code is available at https://github.com/xiaohangt/RMDO. We conducted empirical assessments on a variety of extensive-form games, including Sequential Blotto (perfect-information extensive-form game), Kuhn Poker with a initial pot 40 for each player, Leduc Poker, Leduc Poker Dummy and Oshi Zumo. Leduc Poker Dummy is of particular interest as the NE of the game has a small support as the actions are duplicated in each infoset (McAleer et al., 2021). Oshi Zumo is a board game in which players must repeatedly bid to push a token off the other side of the board (Buro, 2004). The full description of games is in Appendix C.

We evaluate the performance with the exploitability in terms of number of infosets visited and wall time measured in seconds. The number of visited infosets refers to the total number of nodes traversed by the algorithm, including those encountered during best response (BR) computation. It is equivalent to the number of touched nodes in the experiment

Figure 3.Comparison of PDO with XDO, CFR and XFP on Leduc Poker, Dummy Leduc Poker and Oshi Zumo. PDO preserves the strength of DO that performing well in games with small-support NE (Dummy Leduc Poker), but remains competitive to the state-of-the-art regret minimization methods in other games.

of Stochastic Regret Minimization (Farina et al., 2020). It is important to note that the difference between the expanded infosets in XDO paper (McAleer et al., 2021) and the visited infosets in the present study is that XDO did not include the infosets during BR computation.

The experiment uses the state-of-the-art exact regret minimizer, CFR+ (Tammelin, 2014), for all double oracle algorithms, and the regret minimization algorithms are initialized with uniform random policies following the default setting. The study begins by analyzing the performance of PDO with various periodicity choices and comparing them with XODO. Furthermore, the algorithm's performance is compared against baselines, including XDO with restricted game solver CFR+, CFR+ itself, and Extensive-form Fictitious Self-play. For more empirical results including the comparison to Linear CFR, please refer to Appendix E.

In Figure 1, we present a comparison between the XODO and PDO with periodicity values of $n = 1, 10, 50, 100$, in terms of exploitability plotted against wall time in seconds.Our results show that in Kuhn poker, PDO algorithms outperform XODO, with all PDO algorithms exhibiting similar performance. Among the PDO algorithms, we found that PDO with periodicity 100 performs slightly better than the other PDO algorithms. In all the other games, PDO algorithms outperformed XODO by a large margin. In Leduc Poker, larger periodicity values led to faster convergence. Among the PDO algorithms, PDO with periodicity values of 50 and 100 performed the best. In Leduc Poker Dummy, PDO with periodicity 50 achieved a small exploitability the fastest. In Oshi Zumo, large periodicity values led to slow decreasing in exploitability at the early stages of training, but reached the least exploitability later on.

We also compare the performance of PDO with XDO in Figure 2. We find that PDO$_{50}$(10) outperforms XDO with a large margin in Sequential Blotto and Large Kuhn Poker. In Figure 3, we investigate the performance of PDO$_{50}$(10) and other baselines. We find that PDO outperforms XDO and Extensive-form Fictitious Self-play (XFP) with a large margin in Leduc Poker, and has a slight improvement over CFR+ in exploitability in the later stages of the training. In Leduc Poker Dummy, PDO outperforms all other algorithms with a large margin starting from the beginning of the training. In Oshi Zumo, PDO has a more stable exploitability curve and a faster convergence in general compared to XDO, with a lower exploitability level than CFR in the later stages of the training.

Our findings suggest that PDO improves the convergence speed of DO methods in different types of games, and that the choice of periodicity can have a significant impact on the performance of PDO. Furthermore, we find that the last-window average strategy converges faster than the overall average strategy when comparing PDO$_{10}$(10) and XODO, likely due to the poor performance of the strategy before the restricted games stop expanding. These results contribute to the ongoing effort to improve the efficiency and effectiveness of DO algorithms in solving large-scale imperfect-information games.

6. Conclusion

This paper proposes the first generic framework for studying the theoretical convergence speed and algorithmic performance of regret-minimization based Double Oracle algorithms. Building upon this framework, we propose the Periodic Double Oracle algorithm which improves the sample complexity. Our numerical simulations demonstrate that PDO achieves superior performance compared to XDO and ODO in extensive-form games. Additionally, PDO exhibits fast convergence in games with small support NE, and remains robust across other games. Overall, our proposed framework and algorithm offer a significant contribution to the field of Double Oracle methods.

A future direction is combining PDO with deep regret-based methods for solving the restricted game (Perolat et al., 2022; McAleer et al., 2023) and finding the BR (Lanctot et al., 2017; McAleer et al., 2022b;a). As for applications of our framework, our framework can also be applied to robust and risk-aware reinforcement learning (Zhang et al., 2020; Lanier et al., 2022; Slumbers et al., 2022). Additionally, it could be applied to solve extensive-form continuous games

(Adam et al., 2021) where k can be large, but the sample complexity of PDO is only linear in k.

7. Acknowledgements

Yaodong Yang is supported in part by the National Key R&D Program of China (20222D0114900). The authors would like to thank the Department of Statistical Science at University College London for providing the computing clusters. This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/T517793/1, EP/W524335/1].

References

Adam, L., Horík, R., Kasl, T., and Kroupa, T. Double oracle algorithm for computing equilibria in continuous games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5070–5077, 2021.

Blum, A. and Monsour, Y. Learning, regret minimization, and equilibria. 2007.

Bosansky, B., Kiekintveld, C., Lisy, V., and Pechoucek, M. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, 51:829–866, 2014.

Brown, N. Equilibrium Finding for Large Adversarial Imperfect-Information Games. PhD thesis, US army, 2020.

Brown, N. and Sandholm, T. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1829–1836, 07 2019. doi: 10.1609/aaai.v33i01.33011829.

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep counterfactual regret minimization. In *International conference on machine learning*, pp. 793–802. PMLR, 2019.

Buro, M. Solving the oshi-zumo game. In *Advances in Computer Games*, pp. 361–366. Springer, 2004.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Dinh, L. C., McAleer, S. M., Tian, Z., Perez-Nieves, N., Slumbers, O., Mguni, D. H., Wang, J., Ammar, H. B., and Yang, Y. Online double oracle. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=rrMK6hYNSx.

Farina, G., Kroer, C., and Sandholm, T. Stochastic regret minimization in extensive-form games. In *International Conference on Machine Learning*, pp. 3018–3028. PMLR, 2020.

Freund, Y. and Schapire, R. E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

Jonasson, J. et al. On the optimal strategy in a random game. *Electronic Communications in Probability*, 9:132–139, 2004.

Koller, D. and Megiddo, N. The complexity of two-person zero-sum games in extensive form. *Games and economic behavior*, 4(4):528–552, 1992.

Koller, D. and Megiddo, N. Finding mixed strategies with small supports in extensive form games. *International Journal of Game Theory*, 25(1):73–92, 1996.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. Monte carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*, pp. 1078–1086, 01 2009.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidsha ei, S., et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Lanier, J. B., McAleer, S., Baldi, P., and Fox, R. Feasible adversarial robust reinforcement learning for underspecified environments. *arXiv preprint arXiv:2207.09597*, 2022.

McAleer, S., Lanier, J., Fox, R., and Baldi, P. Pipeline PSRO: A scalable approach for finding approximate nash equilibria in large games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

McAleer, S., Lanier, J., Baldi, P., and Fox, R. XDO: A double oracle algorithm for extensive-form games. *Advances in Neural Information Processing Systems (NeurIPS)* 2021.

McAleer, S., Lanier, J., Wang, K., Baldi, P., Fox, R., and Sandholm, T. Self-play psro: Toward optimal populations in two-player zero-sum games. *arXiv preprint arXiv:2207.06541*, 2022a.

McAleer, S., Wang, K., Lanctot, M., Lanier, J., Baldi, P., and Fox, R. Anytime optimal psro for two-player zero-sum games. *arXiv preprint arXiv:2201.07700*, 2022b.

McAleer, S., Farina, G., Lanctot, M., and Sandholm, T. Escher: Eschewing importance sampling in games by computing a history value function to estimate regret. International Conference on Learning Representations, 2023.

McMahan, H. B., Gordon, G. J., and Blum, A. Planning in the presence of cost functions controlled by an adversary. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 536–543, 2003.

Muller, P., Omidsha ei, S., Rowland, M., Tuyls, K., Perolat, J., Liu, S., Hennes, D., Marris, L., Lanctot, M., Hughes, E., et al. A generalized training approach for multiagent learning. In International Conference on Learning Representations, 2019.

Nash Jr, J. F. Equilibrium points in n-person games. Proceedings of the national academy of sciences, 36(1):48–49, 1950.

Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., de Boer, V., Muller, P., Connor, J. T., Burch, N., Anthony, T., et al. Mastering the game of stratego with model-free multiagent reinforcement learning. Science, 378(6623): 990–996, 2022.

Schmid, M., Moravcik, M., and Hladik, M. Bounding the support size in extensive form games with imperfect information. In Proceedings of the Twenty-Eighth AAAI Conference on Arti cial Intelligence, AAAI'14, pp. 784–790. AAAI Press, 2014.

Slumbers, O., Mguni, D. H., McAleer, S., Wang, J., and Yang, Y. Learning risk-averse equilibria in multi-agent systems. arXiv preprint arXiv:2205.15434, 2022.

Tammelin, O. Solving large imperfect information games using cfr+. 07 2014.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, 575 (7782):350–354, 2019.

Von Neumann, J. and Morgenstern, O. Theory of games and economic behavior, 2nd rev. 1947.

Wikipedia. Kuhn poker — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Kuhn%20poker&oldid=1138625516, 2023. [Online; accessed 18-March-2023].

Wilson, R. Computing equilibria of two-person games from the extensive form. Management Science, 18(7):448–460, 1972.

Zhang, H., Chen, H., Boning, D. S., and Hsieh, C.-J. Robust reinforcement learning on state observations with learned optimal adversary. In International Conference on Learning Representations, 2020.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, volume 2008, 01 2007.

A. Additional Details of Extensive-form Double Oracle

A.1. Example of two iterations running

Figure 4. An example of two iterations of Extensive-form Double Oracle. Square player is unaware of what Circle player chose. The values at the leafs are Circle player's payoff. Since it is zero-sum game, the Square player's payoffs are the opposite numbers.

Here we present an illustration of two iterations of the Extensive-form Double Oracle in a zero-sum game in the figure above. In Iteration 0, both Circle and Square players have empty population and employ a uniform random strategy. In Iteration 1, Circle player's BR to Square's random strategy is action L since L and R's expected values are 1.5 and 1, respectively. Square player's BR is action Y since X and Y's expected utilities are 2.5 and 0. Then L and Y are added to the population (thicker lines). Then since both players only have one action in the restricted game, the restricted game's NE is to choose them. Circle player's BR to NE of restricted game is still action L given Square player will choose Y. Meanwhile, Square player's BR is still Y given Circle player will choose L, since Square player's expected values of actions are $v(X) = 2$ and $v(Y) = 1$. As there is no BR actions added to the population, the restricted game won't change in latter iterations, DO converges and the NE for the original game is (L, Y).

A.2. Additional details of Algorithm 1

Algorithm 1 has more detailed than the algorithm description of XDO in the original paper (McAleer et al., 2021). In our description of XDO, we specify according to XDO's official implementation https://github.com/indylab/tabular_xdo/blob/main/main_experiments.py#L173, where the authors start at a initial threshold $= {}_0$ and divide it by 2 every time reaching $-$NE in restricted game.

A.3. Empirical values of k

Table 1 presents the findings of the empirical evaluation of the algorithmic execution of XDO across various games. It is observed that although the value of k is usually smaller than the upper bound $|S|$, even for the smallest $k = 16$, there is a considerable rise in both the number of iterations and the sample complexity caused by the exponential term: 4^{16} $O(10^9)$. The authors of XDO have proposed a solution to address this issue by setting a relatively large threshold value ${}_0$. Nonetheless, the determination of the appropriate value remains a complex task since the sample complexity is highly sensitive to the value of k, which may vary based on the size of the NE support in different games.

Table 1. The empirical values of k when executing XDO to reach 10^{-3}-NE and the number of infosets of different games

GAME NAME	k	NUM. OF INFOSETS
KUHN POKER	20	58
LEDUC POKER	16	9457
LEDUC POKER DUMMY	17	468517
OSHI ZUMO	34	60533

B. Proofs

B.1. Proof of Lemma 3.1

Proof. We first prove the upper bound. The number of time windows equals to the number of times adding new BR actions to the population. Then since the size of the population is less or equal to $|S_i|$ and in each time window there is at least

one new BR action added comparing to the last window, $\sum_i |S_i|$.

Then we prove the lower bound of k. Since there is at most one pure strategy at a infoset added to the population every time when the restricted game is expanded and a new window starts. That is, $|\{a|a \in A(s)\}| \le k$. Since at every infoset, the number of pure strategies in the converged population will be greater than the support of NE. Otherwise, there is a pure strategy in the NE strategy but not in the population, which means that the population doesn't converge and leads to contradiction. At $s \in S$, denote as a set of NE of this game, $|\{a|a \in A(s)\}| \ge \min_{\sigma^* \in \Sigma^*} |\text{supp}(\sigma^*(s))|$. So we have $k \ge \max_{s \in S} |\{a|a \in A(s)\}| \ge \max_{s \in S} \min_{\sigma^* \in \Sigma^*} |\text{supp}(\sigma^*(s))|$. □

B.2. Proof of Lemma 2.2

Proof. Since the weighted-average strategy at time T is defined as $\bar{\sigma}_i^T = \sum_{t=1}^T \frac{w_t}{W_t} \sigma_i^t$, where $\sum_t W_t = 1$. Specially, in Discounted CFR, $W_t = w_t / \sum_t w_t$ and $\lim_{t \to \infty} W_t = 1$. Since the value of strategy $v_i(\cdot, \cdot)$ is a linear function, then we have:

$$\max_{\sigma_i'} \sum_{t=1}^T [v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t)] W_t = \max_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) - v_i(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T) \le O(|S_i| \sqrt{|A_i|/T}) \tag{12}$$

$$\le O(|S_i| \sqrt{|A|/T}) \tag{13}$$

Given the above inequality exists for both i, then in two-player zero-sum game setting, we have:

$$v_i(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T) = -v_{-i}(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T), \max_{\sigma_i'} v_{-i}(\sigma_i', \bar{\sigma}_{-i}^T) = -\min_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T)$$

and then

$$v_i(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T) - \min_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) \le O(|S_{-i}| \sqrt{|A|/T}). \tag{14}$$

Therefore:

$$\begin{aligned} \max_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) - \sum_i O(|S_i| \sqrt{|A|/T}) &\le v_i(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T) \le \max_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) + O(|S_i| \sqrt{|A|/T}) \\ \min_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) - \sum_i O(|S_{-i}| \sqrt{|A|/T}) &\le -v_{-i}(\bar{\sigma}_i^T, \bar{\sigma}_{-i}^T) \le \min_{\sigma_i'} v_i(\sigma_i', \bar{\sigma}_{-i}^T) + \sum_i O(|S_{-i}| \sqrt{|A|/T}); \end{aligned} \tag{15}$$

is ϵ_T-NE, where $\epsilon_T = \sum_i O(|S_i| \sqrt{|A|/T}) = O(|S| \sqrt{|A|/T})$. □

B.3. Proof of Theorem 3.3

Proof. Since iteration 0 population directly add BR with respect to the uniform random strategy, $|A_{i,0}| = 1$, where $|A_{i,j}|$ denotes player i's maximal number of actions over all infoset in time window T_j. Since we have k time windows in the period T, we then have $\sum_{j=0}^{k-1} |T_j| = T$. w_t is the within window weights. In this time window T_j, at most 1 pure action is added to the action set in an infoset compared to the previous time window, then the maximum number of actions at infosets of time window T_j will be bounded by:

$$|A_{i,j}| \le j \le k.$$

Denote V_j as the set of iterations after last BR checking point in T_j. In the last window T_{k-1}, according to assumption 3.2, we have $V_j = \emptyset$ since $\lim_{T \to \infty} |T_{k-1}| = \infty$, there is no "last" since the final window will not end.

Note that since the population will stop growing Every time after computing the BR, RMDO will check if the BR to the average strategies of the opponent is in the current population. Therefore at two different iterations in the same time window, since the populations, we have the local Best Response is exactly the global Best response:

$$\max_{\sigma_i' \in \Pi_j} \sum_{t \in T_j \cap V_j} w_t \, v_i(\sigma_i', \sigma_{-i}^t) = \max_{\sigma_i' \in \Pi_0} \sum_{t \in T_j \cap V_j} w_t \, v_i(\sigma_i', \sigma_{-i}^t). \tag{16}$$

Since the frequency of BR computing in current window T_j is $m(j)$, then $|V_j| < m(j)$ for $j < k-1$. Given the regret minimizer has $O(|S_i|\sqrt{|A_i|T^{-1}})$ (weighted) average regret, then in window T_j for $j = 0, 1, \ldots, k$, we have:

$$\max_{\sigma_i' \in \Sigma_i} \sum_{t \in T_j \cap V_j} w_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \le O\left(|S_i| \sqrt{|A_i^{T_j}|\, |T_j \cap V_j|^{-1}}\right) \le O\left(|S_i| \sqrt{k\, |T_j \cap V_j|^{-1}}\right). \tag{17}$$

Thus:

$$\max_{\sigma_i' \in \Sigma_i} \sum_{t \in T_j \cap V_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \le \frac{|T_j|}{T} O\left(|S_i| \sqrt{k\, |T_j \cap V_j|^{-1}}\right) \tag{18}$$

Then the weighted-average regret:

$$R_i^T = \max_{\sigma_i' \in \Sigma_i} \sum_{t=1}^{T} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma^t) \right) = \sum_j \left[\max_{\sigma_i' \in \Sigma_i} \sum_{t \in T_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \sum_{t \in T_j} W_t \left(v_i(\sigma_i^t, \sigma_{-i}^t) \right) \right] \tag{19}$$

$$= \sum_j \left[\max_{\sigma_i' \in \Sigma_i} \sum_{t \in T_j \cap V_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \sum_{t \in T_j \cap V_j} W_t \left(v_i(\sigma_i^t, \sigma_{-i}^t) \right) + \max_{\sigma_i' \in \Sigma_i} \sum_{t \in V_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \sum_{t \in V_j} W_t \left(v_i(\sigma_i^t, \sigma_{-i}^t) \right) \right] \tag{20}$$

$$= \sum_j \left[\max_{\sigma_i' \in \Sigma_i} \sum_{t \in T_j \cap V_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \sum_{t \in T_j \cap V_j} W_t \left(v_i(\sigma_i^t, \sigma_{-i}^t) \right) + \max_{\sigma_i' \in \Sigma_i} \sum_{t \in V_j} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma_i^t, \sigma_{-i}^t) \right) \sum_{t \in V_j} W_t \left(v_i(\sigma_i^t, \sigma_{-i}^t) \right) \right] \tag{21}$$

$$\sum_j \frac{|T_j|}{T} O\left(|S_i| \sqrt{k\, |T_j \cap V_j|^{-1/2}} + \sum_{t \in V_j} w_t A\right) < \sum_j \frac{|T_j|}{T} O\left(|S_i| \sqrt{k\, |T_j \cap V_j|^{-1/2}} + \sum_{t \in V_j} 1 A\right) \tag{22}$$

$$\frac{1}{T} O\left(|S_i| \sqrt{k} \sum_{j=0}^{k-1} \sqrt{\frac{|T_j|}{|T_j| - m(j) + 1}} + \sum_{j=0}^{k-2} |T_j|[m(j) - 1] A\right). \tag{23}$$

The last but one inequality exists because $|T_j \cap V_j|^{-1/2} \le (|T_j| - m(j) + 1)^{-1/2}$, $V_{k-1} = \emptyset$ and $|V_j| \ge m(j) - 1$.

Therefore, the upper bound of weighted-average regret:

$$R_i^T = \max_{\sigma_i' \in \Sigma_i} \sum_{t=1}^{T} W_t \left(v_i(\sigma_i', \sigma_{-i}^t) - v_i(\sigma^t) \right) < O\left(\sum_{j=0}^{k-2} \frac{|T_j|[m(j) - 1]}{T} + \frac{|S_i| \sqrt{k} \sum_{j=0}^{k-1} \sqrt{\frac{|T_j|}{|T_j| - m(j) + 1}}}{T} A \right). \tag{24}$$

According to Assumption 3.2, $\lim_{T \to \infty} |T_{k-1}|/T = 1$ and $\lim_{T \to \infty} \sum_{j=0}^{k-2} |T_j|/T = 0$. Then we have:

$$\lim_{T \to \infty} m(k-1)/|T_{k-1}| = \lim_{T \to \infty} m(k-1)/T \cdot \lim_{T \to \infty} T/|T_{k-1}| = 0 \cdot 1 = 0 \tag{25}$$

Then if $m(j)$ is sublinear, meaning that $\forall j$, $\lim_{T \to \infty} m(j)/T = 0$, then the weighted-average regret satisfies that:

$$\lim_{T \to \infty} \frac{1}{T} O\left(\sum_{j=0}^{k-2} |T_j|[m(j) - 1] + |S_i| \sqrt{k} \sum_{j=0}^{k-1} \sqrt{\frac{|T_j|}{|T_j| - m(j) + 1}} A \right) \tag{26}$$

$$\lim_{T \to \infty} \frac{1}{T} O\left(\sum_{j=0}^{k-2} |T_j|[m(j) - 1] \right) + \lim_{T \to \infty} \frac{1}{T} O\left(\frac{|S_i| \sqrt{k} \sqrt{|T_{k-1}|}}{\sqrt{1 - m(k-1)/|T_{k-1}|} + 1} \right) + \tag{27}$$

$$\lim_{T \to \infty} \frac{1}{T} O\left(|S_i| \sqrt{k} \sum_{j=0}^{k-2} \sqrt{|T_j|} \right) = \lim_{T \to \infty} \frac{1}{T} O\left(|S_i| \sqrt{k} \sqrt{|T_{k-1}|} \right) = 0. \tag{28}$$

□

B.4. Proof of Theorem 3.4

Proof. Before reaching the global ϵ-NE, in each window T_j, $j < k - 1$, the number of iterations must satisfy that $|T_j| < O(|A||S|^2/\epsilon^2 + m(j) - 1)$ if the regret minimizer has $O(\sqrt{|A||S|/T})$ regret. Since if $|T_j| \geq O(|A||S|^2/\epsilon^2 + m(j) - 1)$, there must be a BR computing happening after $O(|A||S|^2/\epsilon^2)$ iterations, and at this iteration $\mathrm{BR}(\sigma^t) \leq 2\epsilon_j$ and σ^t has reached restricted game's local ϵ-NE, then σ^t is the ϵ-NE in the original game. Besides, in T_{k-1}, we also need less than $O(|A||S|^2/\epsilon^2 + m(j) - 1)$ iterations to reach ϵ-NE. Then sum up iterations in all time window, we get that the expected number of iterations is less than $O(k|A||S|^2/\epsilon^2 - k + \sum_{j=0}^{k-1} m(j))$. □

B.5. Proof of Theorem 4.2

Proof. When finding ϵ-NE, XODO needs $O(|S|^2k^2/\epsilon^2)$ number of iterations and thus computing $O(|S|^2k^2/\epsilon^2)$ times of BR. Given in the worst case, the complexity of traverse the entire game tree once and computing BR is $|S|$, both overall sample complexity is $O(|S|^2k^2/\epsilon^2) \cdot |S| + O(|S|^2k^2/\epsilon^2) \cdot |S| = O(2|S|^2k^2/\epsilon^2)$. □

B.6. Proof of Corollary 4.3

In XDO, when the stopping threshold of restricted games is divided by 1 at the end of each window, XDO needs the following number of iterations to reach ϵ-NE $O(k|A||S|^2/\epsilon^2 + |A||S|^2 \cdot 2^{2k}/\epsilon_0^2 - k)$ ($\epsilon = \epsilon_0/2$ in the original XDO).

Proof. Given in each time window T_j, $m(j) = |A||S|^2 \cdot 2^{2k}/\epsilon_0^2$, plug it into Theorem 3.4, we have the required number of iterations to reach ϵ-NE is $O(k|A||S|^2/\epsilon^2 - k + |A||S|^2(2^{2k} - 1)/[(\epsilon^2 - 1)\epsilon_0^2]) = O(k|A||S|^2/\epsilon^2 + |A||S|^2 \cdot 2^{2k}/\epsilon_0^2])$. □

B.7. Proof of Theorem 4.4

Proof. When finding ϵ-NE, XDO needs $O(k|A||S|^2/\epsilon^2 + |S|4^k/\epsilon_0^2 - k)$ number of iterations. Before convergence, XDO expands the restricted game k times and manage to expand every time when it computes BR. Similarly, given in the worst case, the complexity of traverse the entire game tree once and computing BR is $|S|$, the overall sample complexity is $O(k|A||S|^2/\epsilon^2 + |A||S|^2 4^k/\epsilon_0^2 - k) \cdot |S| = O(k|A||S|^3/\epsilon^2 + |A||S|^3 4^k/\epsilon_0^2)$. □

B.8. Proof of Theorem 4.6

Proof. When finding ϵ-NE, PDO needs $O(k|A||S|^2/\epsilon^2 + (c-1)k)$ number of iterations and thus computing $O(k|A||S|^2/c\epsilon^2 + (c-1)k/c)$ times of BR. Given in the worst case, the complexity of traverse the entire game tree once and computing BR is both $|S|$, the overall sample complexity is

$$O(k|A||S|^2/\epsilon^2 + (c-1)k + k|A||S|^2/c\epsilon^2 + (c-1)k/c) \cdot |S| \tag{29}$$

$$= O(k|A||S|^3/\epsilon^2 + ck|S| + k|A||S|^3/c\epsilon^2 - k|S|/c). \tag{30}$$

□

C. Description of the Games

Sequential Blotto is a revised sequential version of discrete Colonel Blotto Game. Each player at the beginning has a set of forces with different strength and need to put them on board in turns for fighting. Specifically, when the parameters is set to $(4, 25)$, there will be 25 forces for each person with strength from 0 to 24 and 4 times of putting forces on board, thus 2 rounds in total. In each round, players choose a force in turns. The battle results equal to the difference between the power of two forces on board. At the end of this round, the forces will be removed from board and start the next round. The payoff will only appear at the end of the game, equal to the summation of all battle results.

Large Kuhn Poker is a variant of Kuhn Poker where a initial pot for each player is 40 chips. Players can bet any remaining amount.

Leduc Poker Dummy is the same as vanilla Leduc Poekr except the actions in each information set are duplicated twice (McAleer et al., 2021).

Oshi Zumo A board game in which players must repeatedly bid to push a token off the other side of the board. (Buro, 2004). In the instance of our experiment, there are two players each of whom is initialized with 4 coins and the token is originally at the center of a board with length $2K + 1$. K in our case is 6. Players need to put their bid on the board from the amount of coins they have, which is at least 1 and M in our setting is 1. Then the player who have chosen the greater number can push the token one step toward its opponent. Then both player needs to remove the number of coins in their bids. The winner is the player successfully push the token off the side of the board of its opponent. Winner will get payoff 1 and the other will get 1.

We then offer some rough analysis on the support information in limited number common games. There are many more common games shown with small Nash support introduced in Table 2 of ODO paper (Dinh et al., 2022).

GAMES	MINIMUM SUPPORT PERCENTAGE	MAXIMUM SUPPORT PERCENTAGE
SEQUENTIAL BLOTTO (N, M)	1=M	100%
KUHN POKER	50%	100%
LEDUC POKER DUMMY	25%	50%

Table 2. Percentage of Nash support in benchmark games, which is the support size of NE divided by the number of available actions at the corresponding infoset. Since at different infoset we have different support size of NE and number of available actions, we offer minimum and maximum support percentage. The minimum support of sequential Blotto's NE will be the similar idea of tic for tac. Given it is perfect information in our setting, the min player will have a pure strategy NE. The NE of Kuhn Poker is computed in (Wikipedia, 2023). In Leduc Poker Dummy, since actions a duplicated once, the maximum support will only half of number of available actions.

D. Additional details of Periodic Double Oracle

Algorithm 4 Periodic Double Oracle

Hyperparameter $m(j) = c$, initial population $_0$, window index $j = 0$.
for $t = 0; \quad ; 1$ do
  Construct restricted game G_t with $_t$.
  Update ^{t} in G_t.
  if $t \mod c = 0$ then
    Compute $\sim_i^t$ with equation (6).
    for $i \in \{1, 2\}$ do
      $_{t+1} = _t \cup \mathrm{BR}_i(\sim^t_i)$.
    end for
    if $_t \neq _{t-1}$ then
      Start new window $j = j + 1$
      Reset strategy $^{t+1}$.
    end if
    Output ^{t}.
  end if
end for

Table 3. The sample complexity of examples of RMDO to reach NE in an extensive-form game where $|S|$ is the number of infosets and $|A| = \max_{s \in S} |A(s)|$.

EXAMPLE	SAMPLE COMPLEXITY	SAMPLE COMPLEXITY IN k
XODO	$O(2\|S\|^3 k^2 / ^2)$	POLYNOMIAL
XDO	$O(k\|A\|\|S\|^3 / ^2 + \|A\|\|S\|^3 4^k / ^2_0)$	EXPONENTIAL
PDO	$O(k\|A\|\|S\|^3 / ^2 + c k\|S\| + k\|A\|\|S\|^3 / c^{\ 2} \quad k\|S\| / c)$	LINEAR

E. Additional Experimental Results

Figure 5.Performance of PDO with different periodicity and XODO in terms of number of infosets. PDO signi cantly outperforms XODO.

Figure 6.Performance on Sequential Blotto in terms of wall time (in seconds).

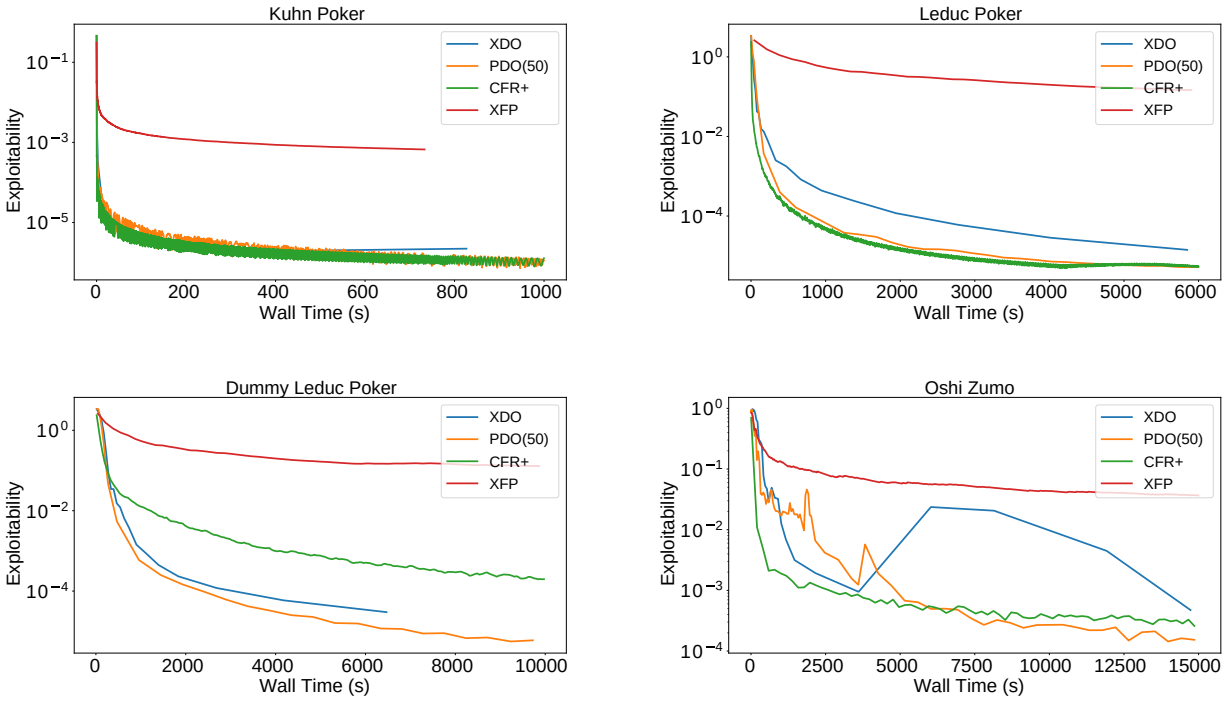

Figure 7. Performance of PDO (50) comparing to baselines.

Figure 8. Performance of PDO with periodicity 50 and other baselines including Linear CFR (exploitability-number of infostates visited).

Figure 9. Performance of PDO with periodicity 50 and other baselines including Linear CFR (exploitability-wall time).