Online Double Oracle

Le Cong Dinh[*,1,2], **Yaodong Yang**[*,1,4], **Stephen McAleer**[5], **Nicolas Perez-Nieves**[3],

Oliver Slumbers[4], **Zheng Tian**[4], **David Henry Mguni**[1], **Haitham Bou Ammar**[1], **Jun Wang**[1,4]

Abstract

Solving strategic games with huge action space is a critical yet under-explored topic in economics, operations research and artificial intelligence. This paper[1] proposes new learning algorithms for solving two-player zero-sum normal-form games where the number of pure strategies is prohibitively large. Specifically, we combine no-regret analysis from online learning with Double Oracle (DO) methods from game theory. Our method – *Online Double Oracle (ODO)* – is provably convergent to a Nash equilibrium (NE). Most importantly, unlike normal DO methods, ODO is *rationale* in the sense that each agent in ODO can exploit strategic adversary with a regret bound of $\mathcal{O}(\sqrt{Tk\log(k)})$ where k is not the total number of pure strategies, but rather the size of *effective strategy set* that is linearly dependent on the support size of the NE. On tens of different real-world games, ODO outperforms DO, PSRO methods, and no-regret algorithms such as Multiplicative Weight Update by a significant margin, both in terms of convergence rate to a NE and average payoff against strategic adversaries.

1 Introduction

Understanding games with large action spaces is a critical topic in a variety of fields including but not limited to economics, operations research and artificial intelligence. A key challenge to solving large games is to compute a Nash equilibrium (NE) [23] in which no player will be better off by deviation. Unfortunately, finding a NE is generally intractable in games; computing a two-player NE is known to be PPAD-hard [9]. An exception is two-player zero-sum games where the NE can be tractably solved by a linear program (LP) [22]. Despite their polynomial-time solvability [30], LP solvers are not adequate in games with prohibitively large action spaces. As a result, researchers shift focus towards finding approximation solutions [21, 17] or developing new solution concepts [26, 32].

Double Oracle (DO) algorithm [21] and its variation Policy Space Response Oracles (PSRO) [17] are powerful approaches to finding approximate NE in games where the support of a NE is relatively small. In the dynamics of DO [21], players are initialised with a limited number of strategies, thus playing only a sub-game of the original game; then, at each iteration, a best-response strategy to the NE of the last sub-game, which is assumed to be given by an *Oracle*, will be added into each agent's strategy pool. The process stops when the best response is already in the strategy pool or the performance improvement becomes trivial. When an exact best-response strategy is not achievable, an approximate solution is often adopted. For example, PSRO methods [17, 20, 25] applies reinforcement learning (RL) [28] oralces to compute the best response.

While DO/PSRO provides an efficient way to approximate the NE in large-scale zero-sum games, it still faces two open challenges. **Firstly**, DO methods require both players to *coordinate* in order to solve the NE in sub-games (i.e., both players have to follow the same learning dynamics such as Fictitious Play (FP) [5]). This contradicts many real-world scenarios where an opponent can play any (non-stationary) strategies in sub-games. **Secondly**, and most importantly, DO methods are not

[1] *Equal contributions. Corresponding to <yaodong.yang@outlook.com>. [1]Huawei U.K. [2] University of Southampton (work done during internship at Huawei). [3]Imperial College London. [4]University College London. [5]University of California, Irvine

Table 1: Properties of existing solvers on two-player zero-sum games $\boldsymbol{A}_{n\times m}$. *:DO in the worst case has to solve all sub-games till reaching the full game, so the time complexity is one order magnitude larger than LP. †: Since PSRO uses approximate best response, the total time complexity is unknown. ‡ Note that the regret bound of ODO can not be directly compared with the time complexity of DO, which are two different notions.

Method	**Rational (No-regret)**	**Allow ϵ-Best Response**	**No Need to Know the Full Matrix A**	**Time Complexity ($\tilde{\mathcal{O}}$) / Regret Bound ($\mathcal{O}$)**	**Large Games**
Linear Programming [30]				$\tilde{\mathcal{O}}\big(n\exp(-T/n^{2.38})\big)$	
(Generalised) Fictitious Play [18]		✓	✓	$\tilde{\mathcal{O}}\big(T^{-1/(n+m-2)}\big)$	
Multipli. Weight Update [12]	✓		✓	$\mathcal{O}\big(\sqrt{\log(n)/T}\big)$	
Double Oracle [21]			✓	$\tilde{\mathcal{O}}\big(n\exp(-T/n^{3.38})\big)^*$	✓
Policy Space Response Oracle [17]		✓	✓	$\times^\dagger$	✓
Online Double Oracle	✓	✓	✓	$\mathcal{O}\big(\sqrt{k\log(k)/T}\big)^\ddagger$	✓

rational [3], in the sense that they do not provide a learning scheme that can exploit the adversary (i.e., achieving no-regret). While a NE strategy guarantees the best performance in the worst scenario, it can be too pessimistic for a player to play such a strategy. For example, in Rock-Paper-Scissors (RPS), playing the NE of $(1/3, 1/3, 1/3)$ makes one player unexploitable. However, if the adversary acts irrationally and addicts to one strategy, say "Rock", then the player should exploit the adversary by consistently playing "Paper" to achieve larger rewards.

No-regret algorithms [7, 27] prescribe a learning scheme in which a player is guaranteed to achieve minimal regret against the best fixed strategy in hindsight when facing an unknown adversary (either rational or irrational). Notably, if both players follow no-regret algorithms, then it is guaranteed that their time-average policies will converge to a NE in zero-sum games [2]. However, the regret bound of popular no-regret algorithms [12, 1] usually depend on the game size; for example, Multiplicative Weight Update (MWU) [12] has a regret bound of $\mathcal{O}(\sqrt{T\log(n)})$ and EXP3 [1] in bandit setting has a regret of $\mathcal{O}(\sqrt{Tn\log(n)})$, where n is number of pure strategies (i.e., experts). As a result, directly applying no-regret algorithms, though rational, cannot solve large-scale zero-sum games.

In this paper, we seek for a scalable solution to two-player zero-sum normal-form games where the game size (i.e., the number of pure strategies) is prohibitively huge. Our main analytical tool is the no-regret analysis in online learning [27]. Specifically, by combining the no-regret analysis [12] with DO methods, we propose *Online Double Oracle (ODO)* algorithm. ODO inherits the key benefits from both sides. It is the first DO method that enjoys the no-regret property and can exploit unknown types of adversary during the game play. Importantly, our ODO method achieves a regret of $\mathcal{O}(\sqrt{Tk\log(k)})$ where k only depends on the support size of the NE rather than the game size. We test our algorithm on tens of games including random matrix games and real-world games of different sizes [10], including Kuhn Poker and Leduc Poker. Results show that in almost all games, ODO outperforms both vanilla DO and strong PSRO [17, 20] and online learning baselines [12] in terms of exploitability (i.e., distance to an NE) and average payoffs against different adversaries.

2 Related Work

The novelty of our ODO methods contributes to both game theory domain and online learning domain. We make the list of existing game solvers for comparisons in Table 1.

Approximating NE has been extensively studied in game theory and multi-agent learning domains [33]. FP [6] and generalised FP [18] are classic solutions where each player adopts a policy that best responds to the time-average policy of the adversary. Although FP is provably convergent to NE in zero-sum games, they are prohibited from solving large games since it suffers from the curse of dimensionality due to the need to iterate all pure strategies in each iteration; and, the convergence rate depends exponentially on the game size [4]. On solving large-scale zero-sum games, DO [21, 19] and PSRO methods [17, 20, 25] have shown remarkable empirical successes. For example, a distributed implementation of PSRO can handle games of size 10^{50} [20]. Yet, both FP and DO methods offer no knowledge about how to exploit the adversary [13], thus regarded as not *rational* [3]. Modern solutions that are rational such as CFR methods [34, 16] are designed for extensive-form games only.

Algorithms with no-(external) regret property can achieve guaranteed performance against the best-fixed strategy in hindsight [27, 7], thus they are commonly applied to tackle adversarial environments. However, conventional no-regret algorithms such as Follow the Regularised Leader [27], Multiplicative Weight Update (MWU) [12] or EXP-3 [1] have the regret bound that is based on the number of pure strategies (i.e., experts). Moreover, these algorithms consider the full strategy set during the update, which deter their applications on large size games. In this paper, we leverage the advantages of both DO methods and no-regret learning algorithms to propose the ODO method. Our ODO enjoys the benefits of both being applicable to solving large games and being able to exploit opponents.

3 Notations & Preliminaries

A two-player zero-sum normal-form game is often described by a payoff matrix $\boldsymbol{A}$ of size $n \times m$. The rows and columns of $\boldsymbol{A}$ are the pure strategies of the row and the column players, respectively. We consider n and m to be prohibitively large numbers. We denote the set of pure strategies for the row player as $\Pi := \{\boldsymbol{a}^1, \boldsymbol{a}^2, \dots \boldsymbol{a}^n\}$, and $C := \{\boldsymbol{c}^1, \boldsymbol{c}^2, \dots, \boldsymbol{c}^m\}$ for the column player. The set of row-player mixed strategies is written as $\Delta_\Pi := \big\{\boldsymbol{\pi} | \boldsymbol{\pi} = \sum_{i=1}^n x_i \boldsymbol{a}^i,\ \sum_{i=1}^n x_i = 1, x_i \geq 0, \forall i \in [n]\big\}$, and for the column player, it is $\Delta_C := \{\boldsymbol{c} | \boldsymbol{c} = \sum_{i=1}^m y_i \boldsymbol{c}^i, \sum_{i=1}^n y_i = 1, y_i \geq 0, \forall i \in [m]\}$. The support of a mixed strategy is written as $\mathrm{supp}(\boldsymbol{\pi}) := \{\boldsymbol{a}^i \in \Pi | x_i \neq 0\}$, with its size $|\mathrm{supp}(\boldsymbol{\pi})|$.

We consider $\boldsymbol{A}_{i,j} \in [0, 1]$ representing the (normalised) loss of the row player when playing against the column player. At the t-th round, the payoff for the joint-strategy profile $(\boldsymbol{\pi}_t \in \Delta_\Pi, \boldsymbol{c}_t \in \Delta_C)$ is written as $(-\boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t, \boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t)$. The row player 's goal is to minimise the quantity $\boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t$. In this paper, we consider the online setting in which players do not know the matrix $\boldsymbol{A}$ and adversary's policy, but rather receive a loss value after the strategy is played: at timestep $t + 1$, the row player observes $\boldsymbol{l}_t = \boldsymbol{A}\boldsymbol{c}_t$ given by the environment, and then it plays a new strategy $\boldsymbol{\pi}_{t+1}$.

Nash Equilibrium. NE of two-player zero-sum games can be defined by the minimax theorem [24]:

$$\min_{\boldsymbol{\pi} \in \Delta_\Pi} \max_{\boldsymbol{c} \in \Delta_C} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c} = \max_{\boldsymbol{c} \in \Delta_C} \min_{\boldsymbol{\pi} \in \Delta_\Pi} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c} = v, \tag{1}$$

for some $v \in \mathbb{R}$. The $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ that satisfies Equation (1) is the NE of the game. In general, one can apply LP method to solve the NE in small games [22]. However, when n and m grows large, the time complexity is not affordable. A more general solution concept is the ϵ-Nash equilibrium, written as

ϵ-Nash Equilibrium. For $\epsilon > 0$, we call a joint strategy $(\boldsymbol{\pi}, \boldsymbol{c}) \in \Delta_\Pi \times \Delta_C$ an ϵ-NE if it satisfies

$$\max_{\boldsymbol{c} \in \Delta_C} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c} - \epsilon \leq \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c} \leq \min_{\boldsymbol{\pi} \in \Delta_\Pi} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c} + \epsilon. \tag{2}$$

Many NE approximation methods (e.g., DO [21] / PSRO [17]) are developed based on the assumption that the support size of NE is small. Formally, it is written as follows.

Assumption 1 (Small Support Size of NEs)**.** *Let $|\cdot|$ denote the cardinality of a set, $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ be a NE of the game $\boldsymbol{A}_{n\times m}$, we assume the support size of $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ is smaller than the game size:.*

$$\max\big(|\,\mathrm{supp}(\boldsymbol{\pi}^*)|, |\,\mathrm{supp}(\boldsymbol{c}^*)|\big) < \min(n, m). \tag{3}$$

Assumption 1 holds in many settings. In symmetric games with random entries [14, Theorem 2.8], it has been proved that the expected support size of NE will be $(\frac{1}{2} + \mathcal{O}(1))n$ where n is the game size; this means the support size of a NE strategy is only half of the game size. In asymmetric games (e.g., $n \gg m$), we provide the following lemma under which Assumption 1 also holds.

Lemma 1. *In asymmetric games $\boldsymbol{A}_{n\times m}, n \gg m$, if the NE $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ is unique, then the support size of the NE will follow $|\,\mathrm{supp}(\boldsymbol{\pi}^*)| = |\,\mathrm{supp}(\boldsymbol{c}^*)| \leq m$. (The full proof is in the Appendix A.1.)*

Notably, the premise of a unique NE in Lemma 1 is a generic property for zero-sum games. On the set of zero-sum normal-form games, the set of zero-sum games with non-unique equilibrium has Lebesgue measure zero [29]. Empirically, we also show that the Assumption 1 holds on tens of real-world zero-sum games [10] (see Table 2 in Appendix C) and randomly generated games (see Figure 1). Later in Section 4.4, we develop solutions when Assumption 1 is violated.

3.1 Double Oracle Method

The pseudocode of DO algorithm [21] is listed in Algorithm 1. The DO method approximates NE in large-size zero-sum games by iteratively creating and solving a series of sub-games (i.e., game with a restricted set of pure strategies). Specifically, at time step t, the DO learner solves the NE of a sub-game $\boldsymbol{G}_t$. Since the sets of pure strategies of the sub-game $\boldsymbol{G}_t = (\Pi_t, C_t)$ are often much smaller than the original game, the NE of sub-game $\boldsymbol{G}_t$ can be easily solved in line 5. Based on the NE of the sub-game: $(\boldsymbol{\pi}_t^*, \boldsymbol{c}_t^*)$, each player finds a best response to NE (line 6), and expand their strategy set (line 7). PSRO methods [17, 20] are variants of DO methods in which RL methods are adopted to approximate the best response strategy. For games where Assumption 1 does not hold, DO methods restore to solve the original game and have no advantages over LP solutions.

Algorithm 1: Double Oracle Algorithm [21]

```
1:  Input: Full strategy set $\Pi, C$
2:  Initialise sets of strategies $\Pi_0, C_0$
3:  for $t = 1$ to $\infty$ do
4:      if $\Pi_t \neq \Pi_{t-1}$ or $C_t \neq C_{t-1}$ then
5:          Solve the NE of the sub-game $\boldsymbol{G}_t$:
            $(\boldsymbol{\pi}_t^*, \boldsymbol{c}_t^*) = \arg\min_{\boldsymbol{\pi} \in \Delta_{\Pi_t}} \arg\max_{\boldsymbol{c} \in \Delta_{C_t}} \boldsymbol{\pi}^\top \boldsymbol{A} \boldsymbol{c}$
6:          Find the best response $\boldsymbol{a}_{t+1}$ and $\boldsymbol{c}_{t+1}$ to $(\boldsymbol{\pi}_t^*, \boldsymbol{c}_t^*)$:
                $\boldsymbol{a}_{t+1} = \arg\min_{\boldsymbol{a} \in \Pi} \boldsymbol{a}^\top \boldsymbol{A} \boldsymbol{c}_t^*$
                $\boldsymbol{c}_{t+1} = \arg\max_{\boldsymbol{c} \in C} \boldsymbol{\pi}_t^{*\top} \boldsymbol{A} \boldsymbol{c}$
7:          $\Pi_{t+1} = \Pi_t \cup \{\boldsymbol{a}_{t+1}\}, C_{t+1} = C_t \cup \{\boldsymbol{c}_{t+1}\}$
8:      else if $\Pi_t = \Pi_{t-1}$ and $C_t = C_{t-1}$ then
9:          Terminate
10:     end if
11: end for
```

Although DO can solve large-scale zero-sum games, it requires to *coordinate* both players to consistently find the NE (see line 5 in Algorithm 1); this renders an disadvantage of DO that when applied in real-world games, it cannot exploit the opponent who can play any non-stationary strategy (see the example of RPS in Introduction). Our ODO solution address this problem by combining DO with tools in online learning.

3.2 Online Learning

Solving the NE in large-scale games is demanding. An alternative approach is to consider learning-based methods. We hope that by playing the same game repeatedly, a learning algorithm could approximate the NE asymptotically. A common metric to quantify the performance of a learning algorithm is to compare its cumulated payoff with the best fixed strategy in hindsight, which is also called *(external) regret*.

Definition 2 (No-Regret Algorithms)**.** *Let $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots$ be a sequence of mixed strategies played by the column player, an algorithm of the row player that generates a sequence of mixed strategies $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots$ is called a no-regret algorithm if we have*

$$\lim_{T \to \infty} \frac{R_T}{T} = 0, \quad R_T = \max_{\boldsymbol{\pi} \in \Delta_\Pi} \sum_{t=1}^{T} \left(\boldsymbol{\pi}_t^\top \boldsymbol{A} \boldsymbol{c}_t - \boldsymbol{\pi}^\top \boldsymbol{A} \boldsymbol{c}_t \right). \tag{4}$$

No-regret algorithms are of great interest in two-player zero-sum games since if both players follow a no-regret algorithm (not necessarily the same one), then the average strategies of both players converge to a NE of the game [7, 2]. For example, a well-known learning algorithm for games that has the no-regret property is the MWU algorithm [12], which is described as

Definition 3 (Multiplicative Weights Update)**.** *Let $\boldsymbol{c}_1, \boldsymbol{c}_2$, ... be a sequence of mixed strategies played by the column player. The row player is said to follow the MWU if $\boldsymbol{\pi}_{t+1}$ is updated as follows*

$$\boldsymbol{\pi}_{t+1}(i) = \boldsymbol{\pi}_t(i) \frac{\exp(-\mu_t {\boldsymbol{a}^i}^\top \boldsymbol{A} \boldsymbol{c}_t)}{\sum_{i=1}^{n} \boldsymbol{\pi}_t(i) \exp(-\mu_t {\boldsymbol{a}^i}^\top \boldsymbol{A} \boldsymbol{c}_t)}, \quad \forall i \in [n] \tag{5}$$

where $\mu_t > 0$ is a parameter, $\boldsymbol{\pi}_0 = [1/n, \ldots, 1/n]$ and n is the number of pure strategies (a.k.a. experts). The regret of MWU can be bounded by $R_T = \max_{\boldsymbol{\pi} \in \Delta_\Pi} R_T(\boldsymbol{\pi}) \leq \sqrt{\frac{T \log(n)}{2}}$.

Intuitively, the MWU method functions by putting larger weights on the experts who have lower losses in the long run. Thus, compared to the best-fixed strategy in hindsight (i.e., the expert with the lowest average loss), the MWU can achieve the no-regret property. However, since the MWU requires updating the whole pure strategy set in each iteration, it is not applicable on large-size games.

4 Online Double Oracle

In this section, we introduce the ODO algorithm. ODO enjoys the no-regret property. Compared to the DO method, ODO can play strategically to exploit the opponent. Compared to the MWU algorithm, ODO can be applied on solving large zero-sum games. The following subsections are organised as follows. We first introduce the key building block of ODO, which is Online Single Oracle (OSO). ODO is the algorithm in which both players adopt the OSO method. We derive ODO's convergence rate to NE, and also analyse ODO's performance when players can only access *approximate* best responses. Finally, we provide a robust extension of OSO considering cases when the size of NE support is unknown (i.e., Assumption 1 may not hold).

4.1 Online Single Oracle Algorithm

One can think of OSO as an online counterpart of *Single Oracle* in DO [21], but achieves no-regret property. A key advantage of OSO is that its regret bound does not depend on a player's size of strategy set, but rather the size of so-called *effective strategy set*, a quantity that is linearly dependent on the size of NE support.

In contrast to classical no-regret algorithms such as MWU [12] where the whole set of pure strategies needs considering at each iteration, i.e., Equation (5), we propose OSO that only considers a *subset* of the whole strategy set. The key operation is that, at each round t, OSO only considers adding a new strategy if it is the best response to the average loss in a time window (defined later). As such, OSO could save exploration cost and computation time by ignoring the pure strategies that have never been the best response to any average losses $\bar{\boldsymbol{l}}$ observed so far, but rather focusing on those effective strategies. This can benefit the learner especially in solving extremely large games.

Algorithm 2: Online Single Oracle Algorithm

```
1: Input: Player's pure strategy set $\Pi$
2: Init. effective strategies set: $\Pi_0 = \Pi_1 = \{\boldsymbol{a}^j\}, \boldsymbol{a}^j \in \Pi$
3: for $t = 1$ to T do
4:     if $\Pi_t = \Pi_{t-1}$ then
5:         Compute $\boldsymbol{\pi}_t$ by the MWU in Equation (5)
6:     else if $\Pi_t \neq \Pi_{t-1}$ then
7:         Start a new time window $T_{i+1}$ and
           Reset $\boldsymbol{\pi}_t = \left[1/|\Pi_t|, \ldots, 1/|\Pi_t|\right]$, $\bar{\boldsymbol{l}} = \mathbf{0}$
8:     end if
9:     Observe $\boldsymbol{l}_t$ and update the average loss in $T_i$:
       $\bar{\boldsymbol{l}} = \sum_{t \in T_i} \boldsymbol{l}_t / |T_i|$
10:    Calculate the best response: $\boldsymbol{a}_t = \arg\min_{\boldsymbol{\pi} \in \Pi} \langle \boldsymbol{\pi}, \bar{\boldsymbol{l}} \rangle$
11:    Update the set of strategies: $\Pi_{t+1} = \Pi_t \cup \{\boldsymbol{a}_t\}$
12: end for
13: Output: $\boldsymbol{\pi}_T, \Pi_T$
```

The pseudocode of OSO is listed in Algorithm 2. We initialise the OSO algorithm with a random subset Π_0 from the original strategy set Π. Without losing generality, we assume that Π_0 starts from only one pure strategy (line 2). We call subset Π_t the **effective strategy set** at time t, and define the period of consecutive iterations as one time window T_i in which the effective strategy set stays fixed, i.e., $T_i := \left\{ t \mid |\Pi_t| = i \right\}$. At iteration t, we update $\boldsymbol{\pi}_t$ (line 5) where only the effect strategy set Π_t (rather than whole set Π) is considered; and the best response is computed against the average loss $\bar{\boldsymbol{l}}$ within the current T_i (line 9). Adding a new best response that is not in the existing effective strategy set will start a new time window (line 7). Notably, despite the design of effective strategy sets, the exact best response of OSO in line 10 still needs considering the whole strategy set Π; we will relax it through approximating best responses in Section 4.3.

We now present the regret bound of OSO in Algorithm 2 as follows.

Theorem 4 (Regret Bound of OSO)**.** *Let $\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_T$ be a sequence of loss vectors played by an adversary, and $\langle \cdot, \cdot \rangle$ be the dot product, OSO in Algorithm 2 is a no-regret algorithm with*

$$\frac{1}{T}\Big(\sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=1}^{T} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle\Big) \leq \frac{\sqrt{k \log(k)}}{\sqrt{2T}},$$

where $k = |\Pi_T|$ is the size of effective strategy set in the final time window.

(We provide the full proof in the Appendix A.2.)

Proof. (of sketch) OSO is designed in such a way that the best response $\boldsymbol{a}_t \in \Pi$ to the average loss must stay in the effective strategy set Π_t; otherwise, a new time window would start. Thus, the best strategy in Π_t must be as good as the best strategy in the whole set Π. Therefore, we can use MWU to bound each time window's regret, resulting in the total regret across all timesteps. □

Remark 1 (The Size of Effective Strategy Set k)**.** *Similar to the original DO, in the worst-case scenario, OSO has to list all pure strategies, i.e., $k = |\Pi|$. However, we believe $k \ll |\Pi|$ holds in many cases. Intuitively, since OSO is a no-regret algorithm, should the adversary follow another no-regret algorithm, the adversary's average strategy would converge to the NE. Thus, learner's effective strategy set with respect to the average loss will include all the pure strategies in the support of learner's NE, which, under Assumption 1, is a far smaller number compared to the game size. Later in Figure 1, we also provide empirical evidence to support the claim of $k \ll |\Pi|$. We believe theoretically upper-bounding k is an important, yet challenging, future work for both DO/PSRO and ODO series of methods.*

OSO with Less-Frequent Best Response. Obtaining a best response strategy can be computationally expensive. It could take hours and days to obtain one single best response [31]. Yet, OSO in Algorithm 2 considers adding a new best-response strategy at every iteration. A practical solution is to consider adding a new strategy when the regret in the current time window exceeds a predefined threshold α. Specifically, if we denote $|\bar{T}_i| := \sum_{h=1}^{i-1} |T_h|$ as the starting time of the time window T_i, and write the threshold at T_i as $\alpha^i_{t-|\bar{T}_i|}$ where $t - |\bar{T}_i|$ denotes the relative position of round t in the time window T_i. We can make OSO add a new strategy only when the following equation is satisfied:

$$\min_{\boldsymbol{\pi}\in\Pi_t} \left\langle \boldsymbol{\pi}, \sum_{j=|\bar{T}_i|}^{t} \boldsymbol{l}_j \right\rangle - \min_{\boldsymbol{\pi}\in\Pi} \left\langle \boldsymbol{\pi}, \sum_{j=|\bar{T}_i|}^{t} \boldsymbol{l}_j \right\rangle \geq \alpha^i_{t-|\bar{T}_i|}. \tag{6}$$

Note that the larger the threshold α, the slower the OSO algorithm adds a new strategy into Π_t. However, choosing a large α will prevent the learner from acquiring the actual best response, thus increasing the total regret R_T by α. In order to maintain the no-regret property, the α needs to satisfy

$$\lim_{T\to\infty} \frac{\sum_{i=1}^{k} \alpha^i_{T_i}}{T} = 0. \tag{7}$$

One choice of α that satisfies Equation (7) can be $\alpha^i_{t-|\bar{T}_i|} = \sqrt{t - |\bar{T}_i|}$. We list the pseudo-code of the OSO under Equation (6) and derive its regret bound of $\mathcal{O}\big(\sqrt{k\log(k)T}\big)$ in Appendix A.3.

4.2 Online Double Oracle

Recall that if both players follow a no-regret algorithm, then the average strategies of both players converge to the NE in two-player zero-sum game [7, 2]. Since OSO algorithm has the no-regret property, it is then natural to study the self-play setting where both players apply OSO (which we call ODO) and investigate its convergence rate to the NE in large games.

Algorithm 3: Online Double Oracle Algorithm

```
Input: Full pure strategy set $\Pi$, $C$
Init. effective strategies set: $\Pi_0 = \Pi_1, C_0 = C_1$
for $t = 1$ to T do
    Each player follows the OSO in Algorithm 2 with
    their respective effective strategy sets $\Pi_t, C_t$
end for
Output: $\boldsymbol{\pi}_T, \Pi_T, \boldsymbol{c}_T, C_T$
```

Compared to the standard DO, ODO no longer needs computing the NE in each sub-game. Furthermore, ODO produces rational agents as each agent can exploit their adversary to achieve the no-regret property. For the ODO method, the convergence result to the NE is presented as follows.

Theorem 5. *Suppose both players apply OSO. Let k_1, k_2 denote the size of effective strategy set for each player. Then, the average strategies of both players converge to the NE with the rate:*

$$\epsilon_T = \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}}.$$

In situation where both players follow OSO with Less-Frequent Best Response in Equation (6) and $\alpha^i_{t-|\bar{T}_i|} = \sqrt{t - |\bar{T}_i|}$, the convergence rate to NE will be

$$\epsilon_T = \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}} + \frac{\sqrt{k_1} + \sqrt{k_2}}{\sqrt{T}}.$$

(We provide the full proof in the Appendix B.1.)

Theorem 5 suggests that similar to OSO, the convergence rate of ODO will not depend on the game size, but rather the size of effective strategy set of both players. As we have illustrated in Remark 1 and also in the later Figure 1, there is a linear relationship between the size of effective strategy set and the support size of the NE. Furthermore, as we will show in Table 2 in Appendix C, in many real games, the size of the NE support is indeed much smaller than the game size. Therefore, our ODO method can be both theoretically and empirically used as a solver in large size zero-sum games.

Online Double Oracle *vs.* Double Oracle with MWU. We want to highlight that ODO is markedly different from simply implementing DO by adopting MWU to solve sub-game NEs (i.e., run MWU till convergence in line 5 in Algorithm 1). Firstly, the MWU update of ODO happens at every iteration and ODO adds a best response per MWU update, whereas DO adds a best response every time a sub-game NE is solved, which often requires thousands of MWU iterations. Most importantly, the best response target in ODO (i.e., the time-average loss $\bar{\boldsymbol{l}}$) is not necessarily a NE; this is in contrast to DO where the best response is computed with respect to an exact NE. In other words, even if DO implements MWU to solve the sub-game NE, it is still not a no-regret algorithm. This also explains the performance gap between ODO and DO (with MWU solving sub-game NEs) in Figure 2.

4.3 Considering ϵ-Best Responses

So far, OSO agents require to compute the exact best response to the average loss function $\bar{\boldsymbol{l}}$ (i.e., line 10 in Algorithm 2). Since calculating the exact best response is often computationally heavy and even infeasible in large games, an alternative way is to consider a ϵ-best response (e.g., through a RL subroutine similar to PSRO [17]). Here we consider the cases where the learner can only access to a ϵ-best response to the average loss. We first present the following lemma showing that by using approximate best response, the original DO can in fact converge to an ϵ-NE in Equation (2).

Lemma 2. *DO will converge to ϵ-NE if players can only access to an ϵ-best response in each round.*

(We provide the full proof in the Appendix B.2.)

Based on Lemma 2, we can now derive the regret bound as well as the convergence guarantee for a OSO learner in the case of ϵ-best response.

Theorem 6. *Suppose OSO agent can only access the ϵ-best response in each iteration when following Algorithm 2, if the adversary follows a no-regret algorithm, then the average strategy of the agent will converge to an ϵ-NE. Furthermore, the algorithm is ϵ-regret:*

$$\lim_{T\to\infty}\frac{R_T}{T}\leq\epsilon;\quad R_T=\max_{\boldsymbol{\pi}\in\Delta_\Pi}\sum_{t=1}^{T}\left(\boldsymbol{\pi}_t^\top\boldsymbol{A}\boldsymbol{c}_t-\boldsymbol{\pi}^\top\boldsymbol{A}\boldsymbol{c}_t\right).$$

(We provide the full proof in the Appendix B.3.)

Theorem 6 implies that in the case of approximate best responses, OSO learners can still approximately converge to a NE. This results essentially authorises the application of optimisation methods to approximate the best response in the ODO process, which paves the way to use RL algorithm in solving complicated zero-sum games such as StarCraft [31].

4.4 Considering Games with Unknown k

The effectiveness of ODO is built on the key condition that the size of the NE support is smaller than the game size (i.e., Assumption 1). Although many real-world zero-sum games do exhibit relatively small k (see Table 2 in Appendix C), in the worst-case scenarios, k can be as large as the game size, in which case one can directly apply a standard no-regret method (e.g., MWU), which we denote as algorithm B. In reality, the size of k is unknown until we solve the game. In the section, we offer a robust variation of ODO that account for the cases when k is unknown (either large or small) and develop a no-regret algorithm that achieves the regret at least as good as ODO and algorithm B.

To fulfil this goal, we can develop a new algorithm by combining OSO with the algorithm B, hoping that the new combined algorithm can achieve the best regret of both algorithms. The Prod algorithm [8] provides an analytical tool to achieve such a goal. Given two algorithms for "easy" and "hard" problem instances, intuitively, the Prod offers a method to combine the two algorithms to achieve the best possible guarantees in both cases. In our setting, the "hard" problem refers to the games with large k while the "easy" problem refers to games with small k.

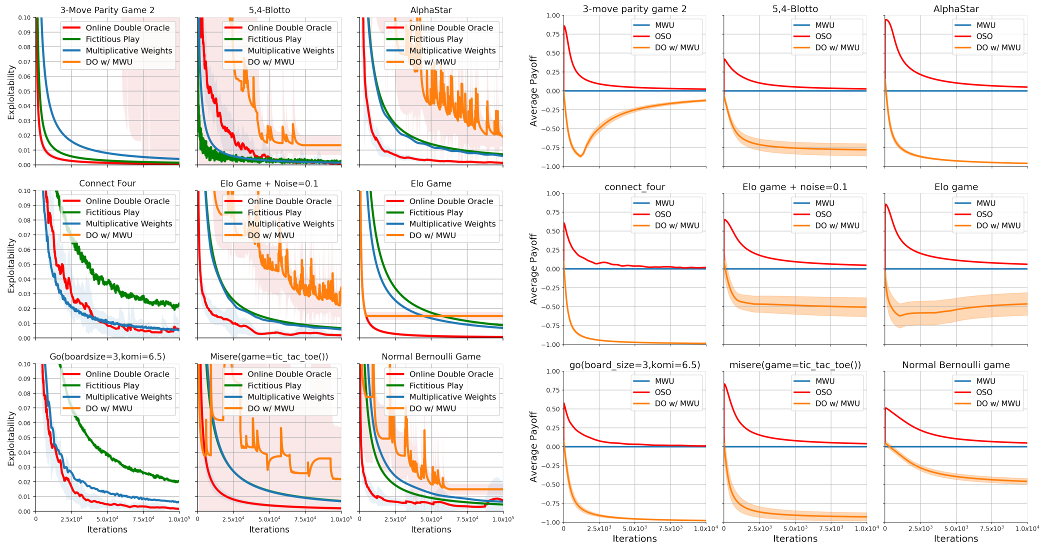
Suppose the no-regret algorithm B has a regret-bound of $R_B(T)$. Then by applying the Prod algorithm, we have the OSO-Prod algorithm. The intuition of OSO-Prod is that, at each iteration t, OSO-Prod updates the weight $w_{t,B}$ for algorithm B based on the relative performance with OSO in round t; if positive, then in the next round $t+1$, by increasing the weight $w_{t,B}$, OSO-Prod will follow strategy of algorithm B with a higher probability. The pseudocode of OSO-Prod is listed in Appendix B.4. Here we provide the regret bound of the OSO-Prod algorithm:

Theorem 7. *Following the OSO-Prod algorithm with the learning rate $\eta=\frac{1}{2}\sqrt{(\log(T)/T)}$ and $w_{1,B}=1-w_{1,A}=1-\eta$ guarantees the regret bound R_T(OSO-Prod) of*

$$\min\left(R_B(T)+2\sqrt{T\log(T)},\frac{T\sqrt{k\log(k)}}{2}+2\log(2)\right).$$

(We provide the full proof in the Appendix B.4.)

By following OSO-Prod, the learner can still achieve a low regret bound when k is small while maintain a regret that is at least as good as the algorithm B in the cases of large k.



(a) Performance comparisons under self-plays (b) Payoffs of playing against an MWU adversary

Figure 2: Performance comparisons on real-world games. Full 30 results are reported in Appendix C

5 Experiments & Results

In this section, we aim to demonstrate the effectiveness of our OSO and ODO methods. Firstly, we verify the small support size of NEs in Assumption 1 through random matrix games. Later, on tens of complicated real-world games, we evaluate our OSO methods in both self-play settings and playing against strategic adversaries. Finally, we show the performance of ODO on large-scale Poker games. As we benchmark on the number of iterations against other baselines, for fair comparisons, we implement the plain OSO in Algorithm 2 without the α threshold trick mentioned in Equation 6 for all our experiments. All hyperparameter settings can be found in Appendix C.

Empirical study on the size of k vs. the full strategy set. We consider a set of zero-sum normal-form games of different sizes, the entries of which are sampled from a uniform distribution $\mathbf{U}(0, 1)$. We run OSO as the row player against a MWU column player until convergence, and plot the size of the OSO player's effective strategy set against its full strategy size. We run 20 seeds for each setting. As we can see from Figure 1, given a fixed support size of NE[2], the size of the effective strategy set k grows as the number of full strategy set increases, but plateaus quickly. The larger the size of NE support (not the full strategy set!), the higher this plateau will reach. Clearly, we can tell that the size of OSO's effective strategy set does not increase with the full strategy size, but rather depends on the support size of a NE. This result confirms Theorem 4 in which we prove that OSO's regret bound depends on k that is related to the size of NE support but not the game size. Economically, this is a desired property as OSO can potentially avoid unnecessary computations in contrast to other no-regret methods that require looping over the full strategy set at each iteration.

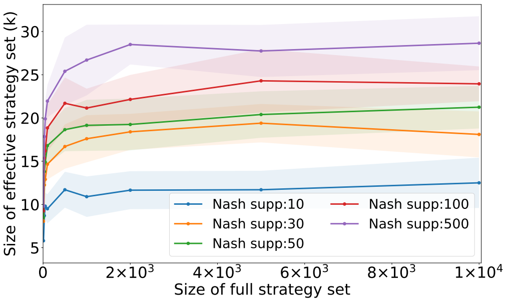
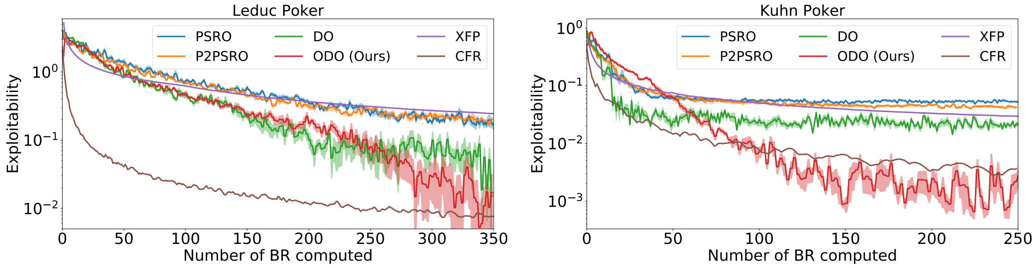


Figure 1: Sizes of effective strategy set (i.e., k) in cases of an OSO agent playing against an MWU opponent with different sizes of full strategy set and NE support.

Performance on Real-World Zero-Sum Games. We investigate ODO in terms of convergence rate to a NE. To demonstrate applicability to real-world problems, we replace the randomly generated normal-form games with 15 popular real-world zero-sum games from Czarnecki et. al. [10]. We compare the *exploitability* [11] (i.e., the distance to a true NE) of ODO with other baseline methods (MWU, FP and DO[3]). We run each game with 20 seeds. As it shows in Figure 2a, ODO outperforms the baselines in almost all 15 games. The advantages of ODO in terms of convergence rate over

[2] We achieve this by fixing the number of columns while increasing the number of rows in the game matrix.

[3] Following the discussion in Section 4.2 and for fair comparisons, we implement DO by adopting MWU as the sub-game NE solver and report the total number of MWU iterations DO needs to achieve a low exploitability.



(a) Exploitability on Leduc Poker

(b) Exploitability on Kuhn Poker

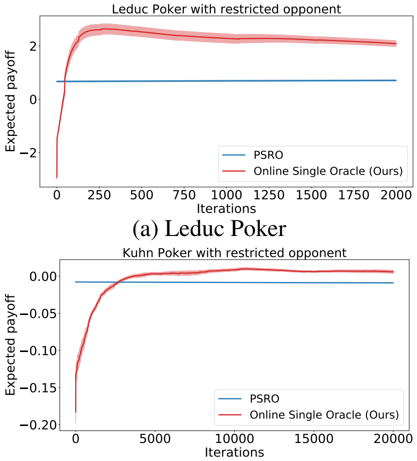
Figure 3: Performance comparisons in exploitability on Poker games.

MWU and FP match our expectation as the support sizes of the NEs in these games are much smaller than the game sizes (reported in Table 2 in Appendix C). For DO with MWU, since it takes many iterations in each sub-game to converge to the NE, it performs much worse than ODO.

Apart from the self-play setting, we also conduct the setting of playing against an MWU adversary in Figure 2b. We can tell that OSO outperforms MWU and DO baselines in average performance in almost all 15 games, which confirms the effectiveness of our design. Notably, MWU achieves a constant payoff; we believe it is because these games are symmetric and since both players follow MWU with the same learning rate, the payoff will always be the value of the game (thus the ground truth), which OSO will eventually converge to as well.

Performance on Poker Games. To further investigate ODO's effectiveness, we test ODO on Kuhn Poker and Leduc Poker. Since ODO is designed only for normal-form games, we adopt the tabular settings [20, 17] in which an exact best response is computed by a tree-traverse oracle (also see OpenSpiel [15]), and for PSRO methods, we perturb the exact best response with random noises. We benchmark how many times such a best-response oracle is called by different methods. We compare against the state-of-the-art PSRO method: P2PSRO[4] [20], and two extensive-form game solvers CFR [34] and XFP [15]. As it shows in Figure 3, ODO shows a significant improvement in exploitability compared to all existing DO and PSRO baselines, and it almost catches up with the state-of-the-art solver CFR in Leduc Poker, and it outperforms CFR on Kuhn Poker. Importantly, ODO uses the less number of best-response calls to achieve the lowest exploitability. We believe they are strongly promising results, since taking the most effective use of best-response functions has a critical impact on solving complex games (e.g., StarCraft [31]).

Can OSO Exploit an *Imperfect* Opponent? Finally, we want to test the no-regret property of OSO when the opponent is *imperfect* and it plays a restricted set. We created such an opponent in both Pokers by restricting its strategy set to only 20 pure strategies, and then let it play MWU. We apply our OSO with ϵ-best response as the row player, and compare its perform against PSRO. We run each setting for 10 random seeds. As we can see from Figure 4, OSO quickly achieves positive expected payoff and outperforms PSRO, which is expected because OSO can actively exploit its opponent while PSRO behaves conservatively by playing NE and not exploiting (thus achieving almost constant payoff). Notably, the average expected payoff of OSO decreases slightly in later iterations, we believe it is because the opponent is following MWU, which is also a no-regret method, and both players will converge towards an NE of such a restricted game.



(a) Leduc Poker

(b) Kuhn Poker

Figure 4: Performance of playing against an imperfect opponent.

6 Conclusion

We propose a new solver for two-player zero-sum games where the number of pure strategies n is huge. Our method, *Online Double Oracle*, absorbs the benefits from both online learning and Double Oracle methods; it achieves the regret bound of $\mathcal{O}(\sqrt{Tk\log(k)})$ where k only depends on the support size of NE rather than n. Unlike DO, ODO can exploit opponents during the game play. In tens of real-world games, we show that ODO outperforms a series of algorithms including MWU, DO and PSRO methods on both convergence rate to NE and average payoff against strategic adversaries.

[4] We have discounted the fact that P2PSRO uses multiple workers (we use two) to compute best responses.

References

[1] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[2] Avrim Blum and Yishay Monsour. Learning, regret minimization, and equilibria. 2007.

[3] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, volume 17, pages 1021–1026. Citeseer, 2001.

[4] Felix Brandt, Felix Fischer, and Paul Harrenstein. On the rate of convergence of fictitious play. In *International Symposium on Algorithmic Game Theory*, pages 102–113. Springer, 2010.

[5] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[6] GW Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation (TC Koopmans, Ed.)*, pages 374–376, 1951.

[7] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[8] Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2):321–352, 2007.

[9] Xi Chen and Xiaotie Deng. Settling the complexity of two-player nash equilibrium. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272. IEEE, 2006.

[10] Wojciech M Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *Advances in Neural Information Processing Systems*, 33, 2020.

[11] Trevor Davis, Neil Burch, and Michael Bowling. Using response functions to measure strategy strength. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[12] Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[13] Sergiu Hart and Andreu Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98(1):26–54, 2001.

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

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

[16] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael H Bowling. Monte carlo sampling for regret minimization in extensive games. In *NIPS*, pages 1078–1086, 2009.

[17] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*, pages 4190–4203, 2017.

[18] David S Leslie and Edmund J Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

[19] Stephen McAleer, John Lanier, Pierre Baldi, and Roy Fox. XDO: A double oracle algorithm for extensive-form games. *Reinforcement Learning in Games Workshop, AAAI*, 2021.

[20] Stephen McAleer, John Lanier, Roy Fox, and Pierre Baldi. Pipeline PSRO: A scalable approach for finding approximate nash equilibria in large games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

[22] Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[23] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[24] J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[25] Nicolas Perez Nieves, Yaodong Yang, Oliver Slumbers, David Henry Mguni, and Jun Wang. Modelling behavioural diversity for learning in open-ended games. *arXiv preprint arXiv:2103.07927*, 2021.

[26] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos. α-rank: Multi-agent evaluation by evolution. *Scientific reports*, 9(1):1–29, 2019.

[27] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

[28] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[29] Eric Van Damme. *Stability and perfection of Nash equilibria*, volume 339. Springer, 1991.

[30] Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

[31] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[32] Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou Ammar. $\alpha\alpha$-rank: Practically scaling α-rank through stochastic optimisation. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1575–1583, 2020.

[33] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[34] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20:1729–1736, 2007.

Supplementary Material for Online Double Oracle

Contents

A Online Single Oracle

A.1 Proof of Lemma 1

Lemma. *In asymmetric games $\boldsymbol{A}_{n\times m}, n \gg m$, if the NE $(\boldsymbol{\pi}^*, \boldsymbol{c}^*)$ is unique, then the support size of the NE will follow $|\operatorname{supp}(\boldsymbol{\pi}^*)| = |\operatorname{supp}(\boldsymbol{c}^*)| \leq m$*

Proof. Since the size of $\boldsymbol{\pi}^*$ and $\boldsymbol{c}^*$ are n and m respectively, the size of the support of NE can not exceed the size of the game:

$$|\text{support}(\boldsymbol{\pi}^*)| \leq n;\ |\text{support}(\boldsymbol{c}^*)| \leq m.$$

In the case the game $\boldsymbol{A}$ has a unique Nash equilibrium, following Theorem 1 in [1], we have:

$$|\text{support}(\boldsymbol{\pi}^*)| = |\text{support}(\boldsymbol{c}^*)| \leq \min(n, m) = m.$$

Thus, we have proved the lemma. □

A.2 Proof of Theorem 4

Theorem (Regret Bound of OSO)**.** *Let $\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_T$ be a sequence of loss vectors played by an adversary, and $\langle \cdot, \cdot \rangle$ be the dot product, OSO in Algorithm 2 is a no-regret algorithm with*

$$\frac{1}{T}\Big(\sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=1}^{T}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle\Big) \leq \frac{\sqrt{k\log(k)}}{\sqrt{2T}},$$

where $k = |\Pi_T|$ is the size of effective strategy set in the final time window.

Proof. W.l.o.g, we assume the player uses the MWU as the no-regret algorithm and starts with only one pure strategy in Π_0 in Algorithm 2. Since in the final time window, the effective strategy set has k elements, there are exactly k time windows. Denote $|T_1|, |T_2|, \ldots, |T_k|$ be the lengths of time windows during each of which the subset of strategies the no-regret algorithm considers does not change. In the case of finite set of strategies, k will be finite and we have

$$\sum_{i=1}^{k}|T_k| = T. \tag{1}$$

In the time window with length $|T_i|$, following the regret bound of MWU in definition 3 we have:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi_{|\bar{T}_i|+1}}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle \leq \sqrt{\frac{|T_i|}{2}\log(i)}, \quad \text{where } |\bar{T}_i| = \sum_{j=1}^{i-1}|T_j|. \tag{2}$$

In the time window T_i, we consider the full strategy set when we calculate the best response strategy in step 11 of Algorithm 2 and it stays in $\Pi_{|\bar{T}_i|+1}$. Therefore, the inequality (2) can be expressed as:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle \leq \sqrt{\frac{|T_i|}{2}\log(i)}. \tag{3}$$

Sum up the inequality (3) for $i = 1, \ldots k$ we have:

$$\sum_{i=1}^{k}\sqrt{\frac{|T_i|}{2}\log(i)} \geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \sum_{i=1}^{k}\min_{\boldsymbol{\pi}\in\Pi}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle$$

$$\geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{i=1}^{k}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle \tag{4a}$$

$$= \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=1}^{T}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle$$

$$\Rightarrow \sqrt{\frac{Tk\log(k)}{2}} \geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=1}^{T}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle. \tag{4b}$$

Inequality (4a) is due to $\sum \min \leq \min \sum$. Inequality (4b) comes from Cauchy-Schwarz inequality and Stirling' approximation. Thus, we have the derived regret. □

A.3 OSO with Less-Frequent Best Response

Algorithm 1: OSO with Less-Frequent Best Response

```
Input: A set $\Pi$ pure strategy set of player
$\Pi_0 := \Pi_1$: initial set of effective strategies;
for $t = 1$ to $\infty$ do
    if $\Pi_t = \Pi_{t-1}$ then
        Following the MWU update in Equation (5)
    else if $\Pi_t \neq \Pi_{t-1}$ then
        Start a new time window $T_{i+1}$
        Reset the MWU update in Equation (5) with a new initial strategy $\boldsymbol{\pi}_t$
    end if
    Observe $\boldsymbol{l}_t$ and update the average loss in the current time window $T_i$
            $\bar{\boldsymbol{l}} = \frac{1}{|T_i|}\sum_{\boldsymbol{\pi}_t \in T_i} \boldsymbol{l}_t$ ;
    Calculate the best response:
            $\boldsymbol{a} = \arg\min_{\boldsymbol{\pi}\in\Pi}\langle\boldsymbol{\pi}, \bar{\boldsymbol{l}}\rangle$,
    if $\min_{\boldsymbol{\pi}\in\Pi_{|\bar{T}_i|+1}}\langle\boldsymbol{\pi}, \sum_{j=|\bar{T}_i|}^{t}\boldsymbol{l}_j\rangle - \min_{\boldsymbol{\pi}\in\Pi}\langle\boldsymbol{\pi}, \sum_{j=|\bar{T}_i|}^{t}\boldsymbol{l}_j\rangle \geq \alpha^i_{t-|\bar{T}_i|}$ then
        Update the strategy set: $\Pi_{t+1} = \Pi_t \cup \boldsymbol{a}$
    else
        $\Pi_{t+1} = \Pi_t$
    end if
    Output the strategy $\boldsymbol{\pi}_t$ at round $t$ for the player
end for
```

Theorem (Regret Bound of OSO with Less-Frequent Best Response)**.** *Let $\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_T$ be a sequence of loss vectors played by an adversary. Then, OSO in Algorithm 1 is a no-regret algorithm with:*

$$\frac{1}{T}\Big(\sum_{t=1}^{T}\langle\boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=1}^{T}\langle\boldsymbol{\pi}, \boldsymbol{l}_t\rangle\Big) \leq \frac{\sqrt{k\log(k)}}{\sqrt{2T}} + \frac{\sum_{i=1}^{k}\alpha^i_{|T_i|}}{T},$$

where $k = |\Pi_T|$ is the size of effective strategy set in the final time window.

Proof. W.l.o.g, we assume the player uses the MWU as the no-regret algorithm and starts with only one pure strategy in Π_0 in Algorithm 2. Since in the final time window, the effective strategy set has k elements, there are exactly k time windows. Denote $|T_1|, |T_2|, \ldots, |T_k|$ be the lengths of time windows during each of which the subset of strategies the no-regret algorithm considers does not change. In the case of finite set of strategies, k will be finite and we have

$$\sum_{i=1}^{k}|T_k| = T.$$

In the time window with length $|T_i|$, following the regret bound of MWU we have:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi_{|\bar{T}_i|+1}}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}, \boldsymbol{l}_t\rangle \leq \sqrt{\frac{|T_i|}{2}\log(i)}, \quad \text{where } |\bar{T}_i| = \sum_{j=1}^{i-1}|T_j|. \tag{5}$$

Since in the time window T_i, the size of the effective strategy set does not change, thus we have:

$$\min_{\boldsymbol{\pi}\in\Pi_{|\bar{T}_i|+1}}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}, \boldsymbol{l}_t\rangle \leq \alpha^i_{|T_i|} \tag{6}$$

From Inequalities (5) and (6) we have:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle\boldsymbol{\pi}, \boldsymbol{l}_t\rangle \leq \sqrt{\frac{|T_i|}{2}\log(i)} + \alpha^i_{|T_i|} \tag{7}$$

Sum up the inequality (7) for $i = 1, \ldots k$ we have:

$$\sum_{i=1}^{k}\left(\sqrt{\frac{|T_i|}{2}\log(i)} + \alpha^i_{|T_i|}\right) \geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \sum_{i=1}^{k}\min_{\boldsymbol{\pi}\in\Pi}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle$$

$$\geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{i=1}^{k}\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle = \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{i=1}^{T}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle \tag{8a}$$

$$\implies \sqrt{\frac{Tk\log(k)}{2}} + \sum_{i=1}^{k}\alpha^i_{|T_i|} \geq \sum_{t=1}^{T}\langle \boldsymbol{\pi}_t, \boldsymbol{l}_t\rangle - \min_{\boldsymbol{\pi}\in\Pi}\sum_{i=1}^{T}\langle \boldsymbol{\pi}, \boldsymbol{l}_t\rangle. \tag{8b}$$

Inequality (8a) is due to $\sum\min \leq \min\sum$. Inequality (8b) comes from Cauchy-Schwarz inequality and Stirling' approximation. Thus, we have the derived regret bound. □

B Online Double Oracle

B.1 Proof of Theorem 5

Theorem. *Suppose both players apply OSO. Let k_1, k_2 denote the size of effective strategy set for each player. Then, the average strategies of both players converge to the NE with the rate:*

$$\epsilon_T = \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}}.$$

In situation where both players follow OSO with Less-Frequent Best Response in Equation (6) and $\alpha^i_{t-|\bar{T}_i|} = \sqrt{t - |\bar{T}_i|}$, the convergence rate to NE will be

$$\epsilon_T = \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}} + \frac{\sqrt{k_1}+\sqrt{k_2}}{\sqrt{T}}.$$

Proof. Using the regret bound of OSO algorithm in Theorem 4 we have:

$$\sum_{t=1}^{T}\boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t - \min_{\boldsymbol{\pi}\in\Pi}\sum_{t=1}^{T}\boldsymbol{\pi}^\top\boldsymbol{A}\boldsymbol{c}_t \leq \sqrt{\frac{Tk_1\log(k_1)}{2}},$$

$$\max_{\boldsymbol{c}\in C}\sum_{t=1}^{T}\boldsymbol{\pi}_t^\top\boldsymbol{A}\boldsymbol{c} - \sum_{t=1}^{T}\boldsymbol{\pi}_t^\top\boldsymbol{A}\boldsymbol{c}_t \leq \sqrt{\frac{Tk_2\log(k_2)}{2}}.$$

From the above inequalities we can derive that

$$\bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\bar{\boldsymbol{c}} \geq \min_{\boldsymbol{\pi}\in\Pi}\boldsymbol{\pi}^\top\boldsymbol{A}\bar{\boldsymbol{c}} \geq \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\pi}_t^\top\boldsymbol{A}\boldsymbol{c}_t - \sqrt{\frac{k_1\log(k_1)}{2T}}$$

$$\geq \max_{\boldsymbol{c}\in C}\sum_{t=1}^{T}\bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\boldsymbol{c} - \sqrt{\frac{k_2\log(k_2)}{2T}} - \sqrt{\frac{k_1\log(k_1)}{2T}}.$$

Similarly, we have

$$\bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\bar{\boldsymbol{c}} \leq \max_{\boldsymbol{c}\in C}\bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\boldsymbol{c} \leq \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\pi}_t^\top\boldsymbol{A}\boldsymbol{c}_t + \sqrt{\frac{k_2\log(k_2)}{2T}}$$

$$\leq \min_{\boldsymbol{\pi}\in\Pi}\boldsymbol{\pi}^\top\boldsymbol{A}\bar{\boldsymbol{c}} + \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}}.$$

Thus, with $\epsilon_T = \sqrt{\frac{k_1\log(k_1)}{2T}} + \sqrt{\frac{k_2\log(k_2)}{2T}}$ we have

$$\max_{\boldsymbol{c}\in C}\sum_{t=1}^{T}\bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\boldsymbol{c} - \epsilon_T \leq \bar{\boldsymbol{\pi}}^\top\boldsymbol{A}\bar{\boldsymbol{c}} \leq \min_{\boldsymbol{\pi}\in\Pi}\boldsymbol{\pi}^\top\boldsymbol{A}\bar{\boldsymbol{c}} + \epsilon_T.$$

By definition, $(\bar{\boldsymbol{\pi}}, \bar{\boldsymbol{c}})$ is ϵ_T-Nash equilibrium.

In situation where both players follow OSO with Less-Frequent Best Response, following Theorem in A.3 we have:

$$\sum_{t=1}^{T} \boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t - \min_{\boldsymbol{\pi}\in\Pi} \sum_{t=1}^{T} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c}_t \leq \sqrt{\frac{Tk_1 \log(k_1)}{2}} + \sum_{i=1}^{k_1} \alpha^i_{|T_i|} \leq \sqrt{\frac{Tk_1 \log(k_1)}{2}} + \sqrt{k1}\sqrt{T}$$

$$\max_{\boldsymbol{c}\in C} \sum_{t=1}^{T} \boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c} - \sum_{t=1}^{T} \boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t \leq \sqrt{\frac{Tk_2 \log(k_2)}{2}} + \sum_{i=1}^{k_2} \alpha^i_{|T_i|} \leq \sqrt{\frac{Tk_2 \log(k_2)}{2}} + \sqrt{k_2}\sqrt{T}.$$

Thus, using the same above arguments in the case of OSO, with $\epsilon_T = \sqrt{\frac{k_1 \log(k_1)}{2T}} + \sqrt{\frac{k_2 \log(k_2)}{2T}} + \frac{\sqrt{k_1}+\sqrt{k_2}}{\sqrt{T}}$, we have:

$$\max_{\boldsymbol{c}\in C} \sum_{t=1}^{T} \bar{\boldsymbol{\pi}}^\top \boldsymbol{A}\boldsymbol{c} - \epsilon_T \leq \bar{\boldsymbol{\pi}}^\top \boldsymbol{A}\bar{\boldsymbol{c}} \leq \min_{\boldsymbol{\pi}\in\Pi} \boldsymbol{\pi}^\top \boldsymbol{A}\bar{\boldsymbol{c}} + \epsilon_T.$$

The proof is complete. □

B.2 Proof of Lemma 2

Lemma. *DO will converge to ϵ-NE if players can only access to an ϵ-best response in each round.*

Proof. We first prove in the case of single oracle algorithm. The double oracle proof will be similar. Since the number of strategies is finite, by the same argument in the the case of exact best response, the process will converge. Suppose that at time step t, the process stops. Since we use ϵ-best response, we have the following relationship:

$$\boldsymbol{\pi}_t{}^\top A_t \boldsymbol{l}_t - \min_{\boldsymbol{\pi}\in\Pi} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{l}_t \leq \epsilon$$

If we set the weight of pure strategies does not appear in $\boldsymbol{\pi}_t$ to be zero to make a $\boldsymbol{\pi}'_t$, then it is obvious that

$$\boldsymbol{\pi}_t^\top A_t = \boldsymbol{\pi}'^\top_t \boldsymbol{A}$$

Thus, we have the following relationship:

$$\boldsymbol{\pi}'^\top_t \boldsymbol{A}\boldsymbol{l}_t - \min_{\boldsymbol{\pi}\in\Pi} \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{l}_t \leq \epsilon \tag{9}$$

Further, since $\boldsymbol{l}_t$ is Nash equilibrium of A_t, we also have

$$\max_{l\in\Delta_l} \boldsymbol{\pi}'^\top_t Al - \boldsymbol{\pi}'^\top_t Al_t = \max_{l\in\Delta_l} \boldsymbol{\pi}_t{}^\top A_t l - \boldsymbol{\pi}_t{}^\top A_t \boldsymbol{l}_t = 0 \tag{10}$$

From inequalities (9) and (10), by definition we conclude that $(\boldsymbol{\pi}'_t, \boldsymbol{l}_t)$ is ϵ-NE of the game $\boldsymbol{A}$. □

B.3 Proof of Theorem 6

Theorem. *Suppose OSO agent can only access the ϵ-best response in each iteration when following Algorithm 2, if the adversary follows a no-regret algorithm, then the average strategy of the agent will converge to an ϵ-NE. Furthermore, the algorithm is ϵ-regret:*

$$\lim_{T\to\infty} \frac{R_T}{T} \leq \epsilon; \quad R_T = \max_{\boldsymbol{\pi}\in\Delta_\Pi} \sum_{t=1}^{T} \left(\boldsymbol{\pi}_t^\top \boldsymbol{A}\boldsymbol{c}_t - \boldsymbol{\pi}^\top \boldsymbol{A}\boldsymbol{c}_t \right).$$

Proof. Suppose that the player uses the Multiplicative Weights Update in Algorithm 2 with ϵ-best response. Denote $|T_1|, |T_2|, \ldots, |T_k|$ be the lengths of time windows during each of which the subset of strategies the no-regret algorithm considers does not change. Furthermore,

$$\sum_{i=1}^{k} |T_k| = T.$$

In a time window T_i, the regret with respect to the best fixed strategy in the effective strategy set is:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{i+1}|} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi_{|\bar{T}_i|+1}} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \sqrt{\frac{|T_i|}{2} \log(i)}, \tag{11}$$

where $|\bar{T}_i| = \sum_{j=1}^{i-1} |T_j|$. Since in the time window T_i, the ϵ-best response strategy stays in $\Pi_{\bar{T}_i+1}$ and therefore we have:

$$\min_{\boldsymbol{\pi} \in \Pi_{|\bar{T}_i|+1}} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \epsilon |T_i|$$

Then, from the inequality (11) we have:

$$\sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \sqrt{\frac{|T_i|}{2} \log(i)} + \epsilon |T_i|, \tag{12}$$

Sum up the inequality (12) for $i = 1, \dots k$ we have:

$$\sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \sum_{i=1}^{k} \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \sum_{i=1}^{k} \sqrt{\frac{|T_i|}{2} \log(i)} + \epsilon |T_i|,$$

$$\Rightarrow \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{i=1}^{k} \sum_{t=|\bar{T}_i|+1}^{|\bar{T}_{(i+1)}|} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \epsilon T + \sum_{i=1}^{k} \sqrt{\frac{|T_i|}{2} \log(i)} \tag{13a}$$

$$\implies \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=1}^{T} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \epsilon T + \sum_{i=1}^{k} \sqrt{\frac{|T_i|}{2} \log(i)}$$

$$\implies \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \min_{\boldsymbol{\pi} \in \Pi} \sum_{t=1}^{T} \langle \boldsymbol{\pi}, \boldsymbol{l}_t \rangle \leq \epsilon T + \sqrt{\frac{T}{2}} \sqrt{k \log(k)}. \tag{13b}$$

Inequality (13a) is due to $\sum \min \leq \min \sum$. Inequality (13b) comes from Cauchy-Schwarz inequality and Stirling' approximation. Using inequality (13b), we have:

$$\min_{\boldsymbol{\pi} \in \Pi} \langle \boldsymbol{\pi}, \bar{l} \rangle \geq \frac{1}{T} \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \sqrt{\frac{k \log(k)}{2T}} - \epsilon. \tag{14}$$

That is, the OSO algorithm is ϵ-regret in the case of ϵ-best response.

Since the adversary follows a no-regret algorithm, we have:

$$\begin{aligned} \max_{l \in \Delta_L} \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, l \rangle - \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle &\leq \sqrt{\frac{T}{2}} \sqrt{\log(L)} \\ \implies \max_{l \in \Delta_L} \langle \bar{\boldsymbol{\pi}}, l \rangle &\leq \frac{1}{T} \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle + \sqrt{\frac{\log(L)}{2T}} \end{aligned} \tag{15}$$

Using the inequalities in (14) and (15) we have:

$$\begin{aligned} \langle \bar{\boldsymbol{\pi}}, \bar{l} \rangle \geq \min_{\boldsymbol{\pi} \in \Pi} \langle \boldsymbol{\pi}, \bar{l} \rangle &\geq \frac{1}{T} \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle - \sqrt{\frac{k \log(k)}{2T}} - \epsilon \\ &\geq \max_{l \in \Delta_L} \langle \bar{\boldsymbol{\pi}}, l \rangle - \sqrt{\frac{\log(L)}{2T}} - \sqrt{\frac{k \log(k)}{2T}} - \epsilon \end{aligned}$$

Similarly, we also have:

$$\begin{aligned} \langle \bar{\boldsymbol{\pi}}, \bar{l} \rangle \leq \max_{l \in \Delta_L} \langle \bar{\boldsymbol{\pi}}, l \rangle &\leq \frac{1}{T} \sum_{t=1}^{T} \langle \boldsymbol{\pi}_t, \boldsymbol{l}_t \rangle + \sqrt{\frac{\log(L)}{2T}} \\ &\leq \min_{\boldsymbol{\pi} \in \Pi} \langle \boldsymbol{\pi}, \bar{l} \rangle + \epsilon + \sqrt{\frac{k \log(k)}{2T}} + \sqrt{\frac{\log(L)}{2T}} \end{aligned}$$

Take the limit $T \to \infty$, we then have:

$$\max_{l \in \Delta_L} \langle \bar{\boldsymbol{\pi}}, l \rangle - \epsilon \leq \langle \bar{\boldsymbol{\pi}}, \bar{l} \rangle \leq min_{\boldsymbol{\pi} \in \Pi} \langle \boldsymbol{\pi}, \bar{l} \rangle + \epsilon$$

Thus $(\bar{\boldsymbol{\pi}}, \bar{l})$ is the ϵ-Nash equilibrium of the game. □

B.4 Prod Algorithm with Online Single Oracle

Denote the OSO Algorithm 2 by A and the existing algorithm with regret bound $R_B(T)$ by B. Then we have the following Prod Algorithm [3]:

Algorithm 2: OSO-Prod algorithm

```
Input: learning rate $\eta \in (0, 1/2]$, initial weights $w_{1,B}, w_{1,A}$, num. of rounds $T$;
for $t = 1, 2, \ldots, T$ do
    Let $s_t = \frac{w_{t,B}}{w_{t,B} + w_{1,A}}$
    Observe $a_t$ and $b_t$ and calculate $x_t = s_t b_t + (1 - s_t) a_t$
    Observe $\boldsymbol{l}_t$ and suffer loss $\boldsymbol{l}_t(x_t)$
    Feed $\boldsymbol{l}_t$ to $A$ and $B$
    Compute $\delta_t = \boldsymbol{l}_t(a_t) - \boldsymbol{l}_t(b_t)$ and set $w_{t+1,B} = w_{t,B}(1 + \eta \delta_t)$
end for
```

Theorem (Theorem 7)**.** *Following the OSO-Prod algorithm with the learning rate* $\eta = \frac{1}{2}\sqrt{(\log(T)/T)}$ *and* $w_{1,A} = 1 - w_{1,B} = 1 - \eta$ *guarantees the regret bound* R_T(*OSO-Prod*) *of*

$$\min(R_T(B) + 2\sqrt{T \log(T)}, \frac{T\sqrt{k \log(k)}}{2} + 2\log(2)).$$

The proof of Theorem 7 comes directly from Corollary 1 in [3]. We provide the Corollary here for the completeness of the paper:

Corollary 1 (Corollary 1 in [3])**.** *Let C* $\geq$ *1 be an upper bound on the total benchmark loss* $L_T(A)$*. Then setting* $\eta = \frac{1}{2}\sqrt{\frac{\log(C)}{C}}$ *and* $w_{1,A} = 1 - w_{1,B} = 1 - \eta$ *simultaneously guarantees*

$$R_T((A, B) - Prod) \leq R_T(B) + 2\sqrt{C \log(C)}$$

and

$$R_T((A, B) - Prod) \leq R_T(A) + 2\log(2).$$

Table 2: Size of the Nash Support of Games

Game	Total Strategies	Size of Nash support
3-Move Parity Game 2	160	1
5,4-Blotto	56	6
AlphaStar	888	3
Connect Four	1470	23
Disc Game	1000	27
Elo game + noise=0.1	1000	6
Elo game	1000	1
Go (boardsize=3,komi=6.5)	1933	13
Misere (game=tic tac toe)	926	1
Normal Bernoulli game	1000	5
Quoridor (boardsize=3)	1404	1
Random game of skill	1000	5
Tic Tac Toe	880	1
Transitive game	1000	1
Triangular game	1000	1

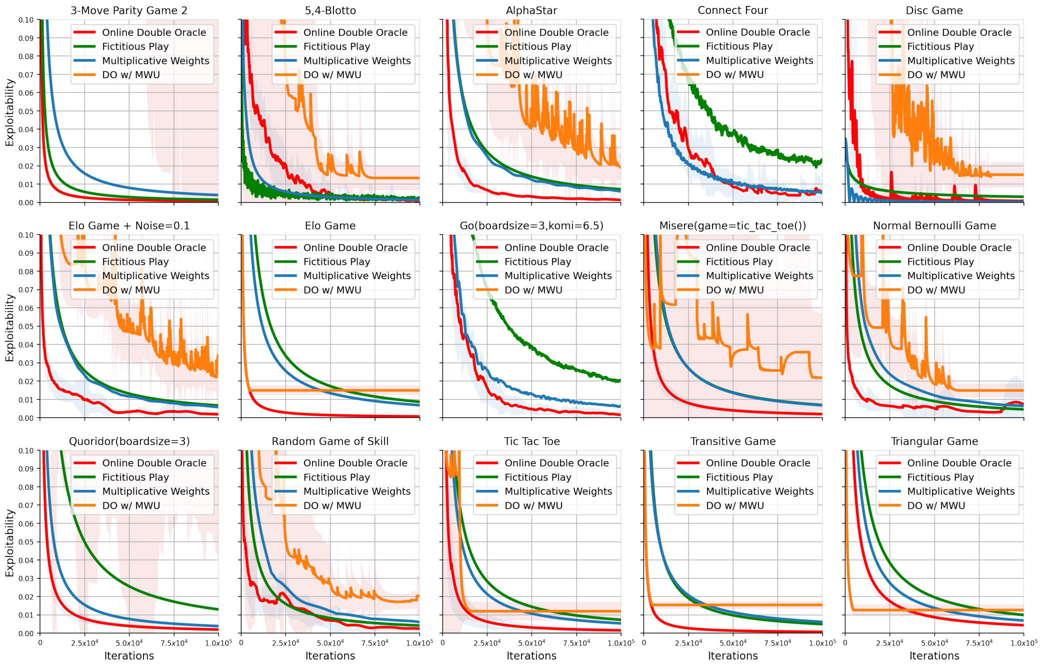


Figure 1: Performance comparisons under self-plays

C Additional Experimental Results

We provide further experiments to demonstrate the performance of the Online Single Oracle and Online Double Oracle algorithms. **All the codes for the experiments are attached as supplementary.**

Performance on Real-World Zero-Sum Games. We report the full results on the 15 popular real-world games on exploitability and average payoff in Figure 1 and Figure 2, respectively.

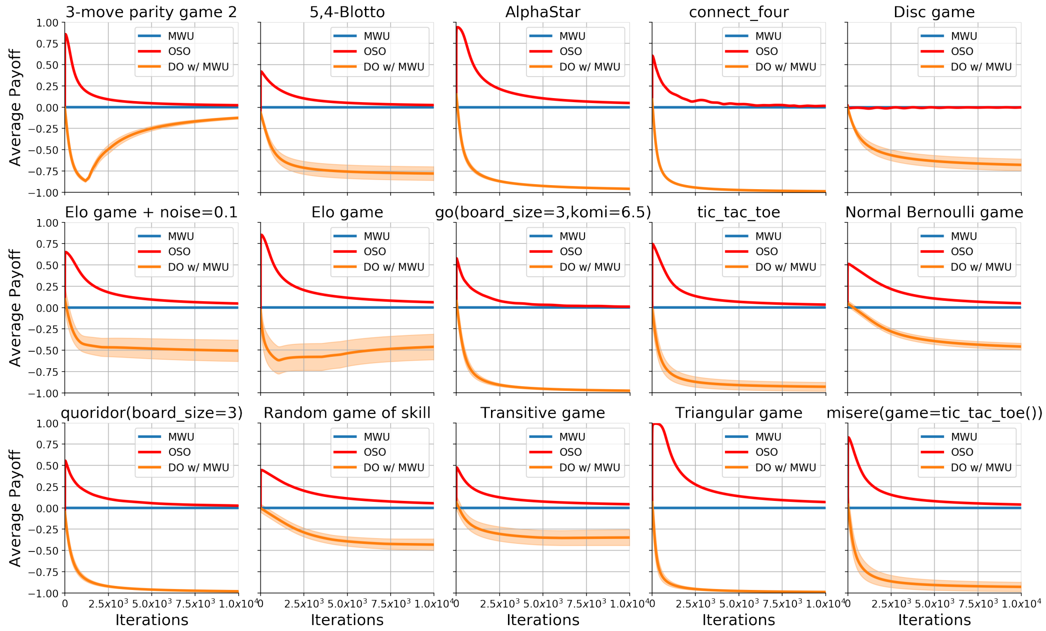


Figure 2: Payoffs of playing against an MWU adversary

Small Support Size of NEs in Real-World Zero-Sum Games. Along with the theoretical guarantee in Lemma 1, we demonstrate that Assumption 1 holds true for many real-world games. To do that, we report the total strategies (i.e., number of pure strategy for each player) and the size of Nash support for 15 popular real-world games [2] in Table 2. As we can see in Table 2, in all of the game we tested, the support size of NEs is a fraction of the game size, reassuring the accuracy of Assumption 1. Notably, in the game with pure NEs (i.e., size of Nash support equals to 1), ODO outperforms other baselines by a large margin, both in exploitability and average payoff.

References

[1] HF Bohnenblust, S Karlin, and LS Shapley. Solutions of discrete, two-person games. *Contributions to the Theory of Games*, 1:51–72, 1950.

[2] Wojciech M Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *Advances in Neural Information Processing Systems*, 33, 2020.

[3] Amir Sani, Gergely Neu, and Alessandro Lazaric. Exploiting easy data in online optimization. In *Advances in Neural Information Processing Systems*, pages 810–818, 2014.