CFR-DO: A Double Oracle Algorithm for Extensive-Form Games

Stephen McAleer[1], **John Lanier**[1], **Pierre Baldi**[1], **Roy Fox**[1]
[1]University of California, Irvine Department of Computer Science
Correspondence to smcaleer@uci.edu

Abstract

Policy Space Response Oracles (PSRO) is a deep reinforcement learning algorithm for two-player zero-sum games that has empirically found approximate Nash equilibria in large games. Although PSRO is guaranteed to converge to a Nash equilibrium, it may take an exponential number of iterations as the number of information states grows. We propose XDO, a new extensive-form double oracle algorithm that is guaranteed to converge to an approximate Nash equilibrium *linearly* in the number of infostates. Unlike PSRO, which mixes best responses at the root of the game, XDO mixes best responses at every infostate. We also introduce Neural XDO (NXDO), where the best response is learned through deep RL. In tabular experiments on Leduc poker, we find that XDO achieves an approximate Nash equilibrium in a number of iterations 1-2 orders of magnitude smaller than PSRO. In experiments on a modified Leduc poker game, we show that tabular XDO achieves over 11x lower exploitability than CFR and over 82x lower exploitability than PSRO and XFP in the same amount of time. We also show that NXDO beats PSRO and is competitive with NFSP on a large no-limit poker game.

Introduction

Policy Space Response Oracles is a deep reinforcement learning method that is based on game theory for finding approximate Nash equilibria (NE) in large two-player zero-sum games. Methods based on PSRO have recently achieved state-of-the-art performance on large imperfect-information two-player zero-sum games such as Starcraft (Vinyals et al. 2019) and Stratego (McAleer et al. 2020). Despite the empirical success of PSRO, in the worst case, PSRO may need to expand all pure strategies in the normal form of the game, which grows exponentially in the number of infostates. The reason for this is that PSRO is based on the Double Oracle algorithm for normal-form games (McMahan, Gordon, and Blum 2003), and a mixture of normal-form pure strategies is an inefficient representation of extensive-form policies.

In this work, we propose a new double oracle algorithm, XDO, that is designed for extensive-form games. XDO keeps a population of pure strategies. At every iteration, XDO creates a restricted game by only considering actions that are chosen by at least one strategy in the population. This restricted game is then approximately solved via an extensive form game solver such as CFR or FP to find a meta-NE, which is extended to the full game by taking arbitrary actions at infostates not encountered in the restricted game. Next, a BR to the restricted game meta-NE is computed via an oracle, and added to the population. XDO can be viewed as a version of PSRO where, instead of solving a restricted game by only mixing population strategies at the root of the game, the algorithm solves the restricted game by mixing population strategies at every infostate.

XDO is guaranteed to converge to an approximate NE in a number of iterations that is linear in the number of infostates, while PSRO may require a number of iterations exponential in the number of infostates. Furthermore, on a worst-case family of games for PSRO, we show that XDO converges in a number of iterations that does not grow with the number of infostates, and grows only linearly with the number of actions at each infostate.

We also introduce a neural version of XDO, called Neural XDO (NXDO). NXDO can be used in games that are large enough to benefit from the generalization over infostates induced by neural-network strategies. NXDO learns an approximate BR through any deep reinforcement learning algorithm. The restricted game is then defined as a meta-game with each meta-action selecting a population policy to play the next action. This restricted game is then solved through any neural extensive-form game solver, such as NFSP or Deep CFR. In our experiments, we use DQN for the approximate BR and NFSP as the restricted game solver.

We conjecture that XDO outperforms PSRO in games where the NE must mix over actions at many infostates. We also conjecture that XDO outperforms CFR in games where the NE only mixes over a small number of actions at each infostate. To demonstrate the effectiveness of our approach on these types of games we run experiments on two games. The first, m-Clone Leduc, is similar to Leduc poker but with every call, fold, and bet action duplicated m times. The second game is a small no-limit poker game with unabstracted integer bet sizes from 0 to 12. We show that tabular XDO greatly outperforms PSRO, CFR, and XFP on m-Clone Leduc. We also show that NXDO outperforms both PSRO and NFSP on m-Clone Leduc, and beats PSRO by a larger margin than NSFP does in the no-limit poker game.

To summarize, our contributions are as follows:

- We present a tabular extensive form double oracle algorithm, XDO, that terminates in a linear number of iterations in the number of infostates.
- We present a neural version of XDO, NXDO, that outperforms PSRO and NFSP on a modified Leduc poker game and beats PSRO while tying with NFSP on a no-limit poker game.

Background

Extensive-Form Games

In this section, we use the same notation as in DREAM (Steinberger, Lerer, and Brown 2020). An extensive-form game progresses through a sequence of player actions, and has a **world state** $w \in \mathcal{W}$ at each step. In an N-player game, $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ is the space of joint actions for the players. $\mathcal{A}_i(w)$ denotes the set of legal actions for player $i \in \mathcal{N} = \{1, \ldots, N\}$ at world state w and $a = (a_1, \ldots, a_N) \in \mathcal{A}$ denotes a joint action. At each world state, after the players choose a joint action, a transition function $\mathcal{T}(w, a) \in \Delta^{\mathcal{W}}$ determines the probability distribution of the next world state w'. Upon transition from world state w to w' via joint action a, player i makes an **observation** $o_i = \mathcal{O}_i(w, a, w')$. In each world state w, player i receives a reward $\mathcal{R}_i(w)$.

A **history** is a sequence of actions and world states, denoted $h = (w^0, a^0, w^1, a^1, \ldots, w^t)$, where w^0 is the known initial world state of the game. $\mathcal{R}_i(h)$ and $\mathcal{A}_i(h)$ are, respectively, the reward and set of legal actions for player i in the last world state of a history h. An **infostate** (information set) for player i, denoted by s_i, is a sequence of that player's observations and actions up until that time point $s_i(h) = (a_i^0, o_i^1, a_i^1, \ldots, o_i^t)$. Define the set of all infostates for player i to be $\mathcal{I}_i$. The set of histories that correspond to an infostate s_i is denoted $\mathcal{H}(s_i) = \{h : s_i(h) = s_i\}$, and it is assumed that they all share the same set of legal actions $\mathcal{A}_i(s_i(h)) = \mathcal{A}_i(h)$.

A player's **policy** $\pi_i \in (\Delta^{\mathcal{A}_i})^{\mathcal{I}_i}$ is a function mapping from an infostate to a probability distribution over actions. A **policy profile** π is a tuple $(\pi_1, \ldots, \pi_N)$. All players other than i are denoted $-i$, and their policies are jointly denoted π_{-i}. A policy for a history h is denoted $\pi_i(h) = \pi_i(s_i(h))$ and $\pi(h)$ is the corresponding policy profile. We also define the transition function $\mathcal{T}(h, a_i, \pi_{-i})$ as a function drawing actions for $-i$ from π_{-i} to form $a = (a_i, a_{-i})$ and to then sample the next world state w' from $\mathcal{T}(w, a)$, where w is the last world state in h.

The **expected value (EV)** $v_i^\pi(h)$ for player i is the expected sum of future rewards for player i in history h, when all players play policy profile π. The EV for an infostate s_i is denoted $v_i^\pi(s_i)$ and the EV for the entire game is denoted $v_i(\pi)$. A **two-player zero-sum** game has $v_1(\pi) + v_2(\pi) = 0$ for all policy profiles π. The EV for an action in an infostate is denoted $v_i^\pi(s_i, a_i)$. A **Nash equilibrium (NE)** is a policy profile such that, if all players played their NE policy, no player could achieve higher EV by deviating from it. Formally, π^* is a NE if $v_i(\pi^*) = \max_{\pi_i} v_i(\pi_i, \pi^*_{-i})$ for each player i.

The **exploitability** $e(\pi)$ of a policy profile π is defined as $e(\pi) = \sum_{i \in \mathcal{N}} \max_{\pi'_i} v_i(\pi'_i, \pi_{-i})$. A **best response (BR)** policy $BR_i(\pi_{-i})$ for player i to a policy π_{-i} is a policy that maximally exploits π_{-i}: $BR_i(\pi_{-i}) = \arg\max_{\pi_i} v_i(\pi_i, \pi_{-i})$. An **$\epsilon$-best response ($\epsilon$-BR)** policy $BR_i^\epsilon(\pi_{-i})$ for player i to a policy π_{-i} is a policy that is at most ϵ worse for player i than the best response: $v_i(BR_i^\epsilon(\pi_{-i}), \pi_{-i}) \geq v_i(BR_i(\pi_{-i}), \pi_{-i}) - \epsilon$. An **$\epsilon$-Nash equilibrium ($\epsilon$-NE)** is a policy profile π in which, for each player i, π_i is an ϵ-BR to π_{-i}.

A **normal-form game** is a single-step extensive-form game. An extensive-form game induces a normal-form game in which the legal actions for player i are its deterministic policies $_{s_i \in \mathcal{I}_i} \mathcal{A}_i(s_i)$. These deterministic policies are called **pure strategies** of the normal-form game. Since each deterministic policy specifies one action at every infostate, there are an exponential number of pure strategies in the number of infostates. A **mixed strategy** is a distribution over a player's pure strategies. Two policies π_i^1 and π_i^2 for player i are said to be **realization-equivalent** if for any fixed strategy profile of the other player, both π_i^1 and π_i^2, define the same probability distribution over the states of the game.

Theorem 1 (Kuhn's Theorem (Kuhn and Tucker 1953))**.** *Any mixed strategy in the normal form of a game is realization equivalent to a policy in the extensive form of that game, and vice versa.*

Related Work

There has been much recent work on non-game-theoretic multi-agent RL (Foerster et al. 2018; Lowe et al. 2017; Rashid et al. 2018; Bansal et al. 2017). Most of this work focuses on games with more than two players such as multi-agent cooperative games or mixed competitive-cooperative scenarios. In cooperative environments, self-play has empirically been shown to find an approximate NE (Lowe et al. 2017; Majumdar et al. 2020), but can be brittle when cooperating with agents it hasn't trained with (Lanctot et al. 2017). Self-play reinforcement learning has achieved expert level performance on video games (Vinyals et al. 2019; Berner et al. 2019; Jaderberg et al. 2019), but is not guaranteed to converge to an approximate NE.

Extensive-form fictitious play (XFP) (Heinrich, Lanctot, and Silver 2015) and counterfactual regret minimization (CFR) (Zinkevich et al. 2008) extend fictitious play and regret matching, respectively, to extensive-form games. Deep CFR (Brown et al. 2019) is a general method that trains a neural network on a buffer of counterfactual values. However, Deep CFR uses external sampling, which may be impractical for games with a large branching factor such as Stratego and Barrage Stratego. DREAM (Steinberger, Lerer, and Brown 2020) and ARMAC (Gruslys et al. 2020) are model-free regret-based deep learning approaches.

Our work is also related to pruning approaches (Brown and Sandholm 2015; Brown, Kroer, and Sandholm 2017). These methods start with all actions and sequentially remove actions that have low expected value. XDO instead starts with no actions and sequentially adds actions.

Neural Fictitious Self Play (NFSP)

Neural Fictitious Self Play (NFSP) (Heinrich and Silver 2016) approximates XFP by progressively training a best response against an average of all past policies using reinforcement learning. The average policy is represented by a neural network and is trained via supervised learning using a replay buffer of past best response actions. Each episode, both players either play from their best response policy with probability $\eta = 0.1$ or with their average policy with probability $1 - \eta$. This experience is then added to the best response circular replay buffer and is used to train the best response for both players with off-policy DQN. If a player plays with their best response policy, the data is also added to the average policy reservoir replay buffer and is used to train the average policy via supervised learning.

Policy Space Response Oracles (PSRO)

The Double Oracle algorithm (McMahan, Gordon, and Blum 2003) is an algorithm for finding a NE in normal-form games. The algorithm works by keeping a population of policies Π^t at time t. Each iteration a NE $\pi^{*,t}$ is computed for the game restricted to policies in Π^t. Then, a best response to this NE for each player $BR_i(\pi^{*,t}_{-i})$ is computed and added to the population $\Pi^{t+1}_i = \Pi^t_i \cup \{BR_i(\pi^{*,t}_{-i})\}$ for $i \in \{1, 2\}$.

Policy Space Response Oracles (PSRO) (Lanctot et al. 2017) approximates the Double Oracle algorithm. The meta NE is computed on the empirical game matrix U^{Π}, given by having each policy in the population Π play each other policy and tracking average utility in a payoff matrix. In each iteration, an approximate best response to the current meta NE over the policies is computed via any reinforcement learning algorithm. Pipeline PSRO parallelizes PSRO with convergence guarantees (McAleer et al. 2020).

A primary issue with PSRO is that it is based on a normal-form algorithm, and the number of pure strategies in a normal-form representation of an extensive-form game is exponential in the number of information sets. Our approach implements the double oracle algorithm directly in the extensive-form game, overcoming this problem and terminating in a linear number of iterations in the number of infostates.

Close to our work, (Bosansky et al. 2014) develop a sequence form double oracle algorithm for extensive-form games that terminates in a linear number of iterations in the number of infostates. Because this method works directly on the sequence form, however, it is not clear how to extend this method to large games with neural networks.

Extensive-Form Double Oracle (XDO)

We propose Extensive-Form Double Oracle (XDO), a double-oracle algorithm designed for two-player zero-sum extensive-form games. XDO maintains a population of pure strategies, and in each iteration computes a meta-Nash Equilibrium (meta-NE) of this population. Then the algorithm finds a best response (BR) to the meta-NE and adds it to the population.

In XDO, the population induces a different restricted game, and therefore a different population meta-NE, than in PSRO. In PSRO, a restricted normal-form game is induced by playing off each pair of population strategies. In XDO, a restricted extensive-form game is induced where the allowed actions in each node are only those suggested by any strategy in the population. This allows the mixed-strategy meta-NE in XDO to assign different weights to the population strategies at every infostate.

XDO uses a tabular method such as CFR to solve the restricted game. The algorithm terminates after an iteration in which the neither of the players finds a BR that outperforms the meta-NE. When this happens, the meta-NE policies are BRs to each other in the original game as well, and the meta-NE is therefore the NE of the original game.

Interestingly, at each but the final iteration of XDO at least one player adds some new action at some non-terminal infostate, because a BR cannot outperform the meta-NE with only restricted game actions. When player i adds action a_i in infostate s_i, this adds to the restricted game previously excluded infostates of the form (s_i, a_i, o_i). The number of iterations that XDO takes to terminate is therefore at most the number of infostates. In contrast, the best known guarantee for the number of iterations that PSRO takes to terminate is exponential in the number of infostates, because PSRO may need to add all pure strategies to the population. Moreover, computing the meta-NE in PSRO may become intractable in later iterations as the population size increases.

Algorithm 1: CFR-DO

```
Result: Approximate Nash Equilibrium
Input: initial population Π^0
while Not terminated do
    Define restricted game via (3);
    Get ϵ-Nash policy π^r of restricted game via CFR;
    Find BR_i(π^r_{-i}) for i ∈ {1, 2} via oracle;
    Π_i^{t+1} = Π_i^t ∪ BR_i(π^r_{-i}) for i ∈ {1, 2};
    if No new actions are added then
        Terminate
    end
end
```

Formally, XDO keeps a population of pure strategies Π^t at time t. Each iteration, a restricted extensive-form game is created and a NE to the restricted game is computed. The restricted game is created by taking the original game and restricting the actions at every infostate s_i to be only the actions where there exists a policy in the population Π^t that chooses that action at that infostate:

$$\mathcal{A}^r_i(s_i) = \{a \in \mathcal{A}_i(s_i) : \exists \pi_i \in \Pi^t \text{ s.t. } \pi_i(s_i, a) = 1\} \quad (1)$$

Then, an ϵ-NE policy π^{r*} is computed in this restricted game via a tabular method such as CFR and is extended to the full game by defining arbitrary actions on infostates not encountered in the restricted game. Next, BRs to this restricted game meta-NE $BR_1(\pi^{r*}_2)$ and $BR_2(\pi^{r*}_1)$ are computed via an oracle. These BRs are then added to the population of policies: $\Pi^{t+1}_i = \Pi^t_i \cup BR_i(\pi^{r*}_{-i})$ for $i \in \{1, 2\}$.

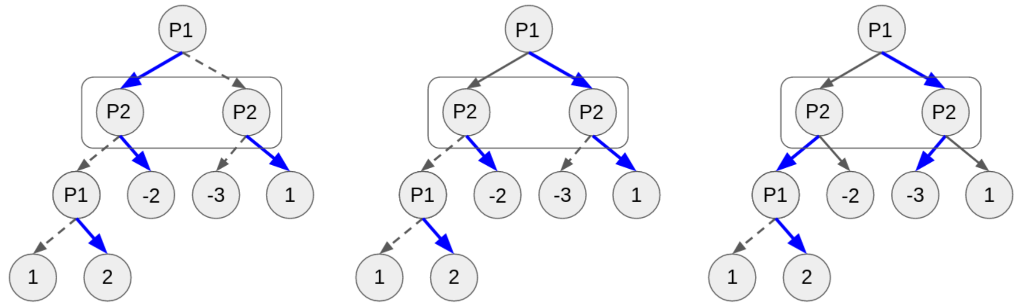

Figure 1: Three iterations of XDO (left to right). In these extensive-form game diagrams, player 1 (P1) plays at the root, then P2 plays without knowing P1's action, and if both played Left P1 plays another action. Actions in the restricted game are solid, vs. dashed outside the restricted game. Meta-NE actions are blue, vs. black not in the meta-NE. BR actions are thick, vs. thin for non-BR actions.

The algorithm terminates when neither player benefits more than ϵ from deviating from the meta-NE to the BR, indicating that the meta-NE is an ϵ-NE also in the original game. To speed up XDO, we start the algorithm with high value of the ϵ parameter that controls the meta-NE approximation quality, and decrease ϵ by half every time the exploitability of the meta-NE in the original game is less than the current ϵ.

To illustrate how XDO works, we demonstrate a simple game in Figure 1. The game starts with empty populations and arbitrary actions (always Left) in every infostate. At the first iteration (left diagram), player 1 adds a BR that plays Left at the first infostate (the root) and Right at the second one. Player2 simultaneously adds a BR that plays Right at their single infostate. The restricted game now consists of only these added actions. At the second iteration (middle diagram), player 1 adds a BR that plays Right at both infostates, and player 2's BR still plays Right. The restricted game now includes both actions for the root infostate, but only Right is in the meta-NE. Next, in the third iteration (right diagram), player 1 keeps the same BR, while player 2's BR plays Left. In the meta-NE of this final restricted game, player 1 plays Left and Right with equal probability at the first infostate, and player 2 plays Left with probability 0.37 and Right with probability 0.63. Since the BRs to this meta-NE do not add any new actions, XDO terminates, and the meta-NE is the NE for the full game. Note that in this example, most actions are needed to find a NE. In games like this, it would be faster to simply solve the original game from the beginning. However, certain games such as the ones in our experiments have Nash equilibria that only need to mix over a small subset of actions, in which case XDO will be much faster than solving the original game.

Proposition 1. *In XDO with an ϵ_1-BR oracle, let π^{r*} be the final ϵ_2-NE in the restricted game. Then π^{r*} is an $(\epsilon_1 + \epsilon_2)$-NE in the full game.*

Proof. For each $i \in \{1, 2\}$, let $BR_i^{\epsilon_1}(\pi_{-i}^{r*})$ be player i's ϵ_1-BR to π_{-i}^{r*} obtained in the last iteration. By the termination condition $v_i(\pi^{r*}) \geq v_i(BR_i^{\epsilon_1}(\pi_{-i}^{r*}), \pi_{-i}^{r*}) - \epsilon_2$ $\geq \max_{\pi_i'} v_i(\pi_i', \pi_{-i}^{r*}) - \epsilon_1 - \epsilon_2$, where the last inequality follows from $BR_i^{\epsilon_1}(\pi_{-i}^{r*})$ being an ϵ_1-best response to π_{-i}^{r*}. □

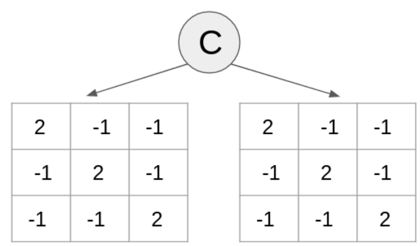

Figure 2: A 2-GMP game with $n = 3$ actions. The chance node selects uniformly at random which GMP game both players play. Both players know what stage game they are playing.

The next two propositions show an exponential gap in the known guarantees for the number of iterations in which PSRO and XDO terminate. If each infostate allows A different actions, PSRO is guaranteed to terminate in $\sum_i A^{|\mathcal{I}_i|}$ iterations, while XDO is guaranteed to terminate in $\sum_i |\mathcal{I}_i|$ iterations.

Proposition 2. *PSRO terminates in $\sum_i \prod_{s_i \in \mathcal{I}_i} \mathcal{A}_i(s_i)$ iterations.*

Proof. In each iteration of PSRO, at least one player adds a new normal-form pure strategy to the population. The space of pure strategies for player i has size $\prod_{s_i \in \mathcal{I}_i} \mathcal{A}_i(s_i)$, because each normal-form pure strategy specifies an action at each infostate for that player. □

Proposition 3. *XDO terminates in $\sum_i |\mathcal{I}_i|$ iterations.*

Proof. Consider an infostate $s_i' = (a_i^0, o_i^1, \ldots, a_i^t, o_i^{t+1})$ for player i as *covered* in the restricted game if any of player i's population policies chooses action a_i^t in infostate $s_i = (a_i^0, o_i^1, \ldots, a_i^{t-1}, o_i^t)$. At each but the final iteration, at least one player i has $v_i(BR_i(\pi_{-i}^{r*}), \pi_{-i}^{r*}) > v_i(\pi^{r*}) + \epsilon$. Since π_i^{r*} is an ϵ-BR to π_{-i}^{r*} in the restricted game, the BR $BR_i(\pi_{-i}^{r*})$ must be choosing at least some action a_i at some non-terminal infostate s_i that was not previously chosen by any population policy. Adding this action to the restricted game covers at least one previously uncovered infostate: all infostates $s'_i = (s_i, a_i, o_i)$, for any observation o_i. All infostates will therefore be covered in at most $\sum_i |\mathcal{I}_i|$ iterations, at which point the next iteration must terminate. □

Tightness of the guarantees. The guarantees in Proposition 2 and Proposition 3 are tight in the sense that they are achieved in some games, but more nuanced analysis is required to identify easier cases where these bounds overestimate the complexity of the algorithms. Both PSRO and XDO often outperform these guarantees and terminate in fewer iterations. A case in which PSRO expands all pure normal-form strategies of an extensive-form game is described in the supplementary materials.

XDO can add multiple actions in each iteration. In practice, XDO often outperforms the guarantee of Proposition 3 because it adds multiple actions in each iteration. Here we present and analyze a family of games in which XDO terminates in asymptotically fewer iterations than suggested by the bound in Proposition 3.

In a generalized matching pennies (GMP) game, both players simultaneously choose one of n actions. The payoff to player 1 is $n-1$ if the actions match, or -1 if they are different. In a k-GMP game (Figure 2), a chance node first selects an index j between 1 and k, and then the players play the j'th of k identical GMP games. The following proposition provides a tighter performance bound for XDO in this case, $2n$ iterations instead of $\sum_i |\mathcal{I}_i| = 2k(n+1)$ (there are kn terminal infostates for each player).

Proposition 4. *In k-GMP with n actions, XDO terminates in $2n$ iterations.*

Proof. In a given iteration, consider the restricted game for a single GMP game. If player 2 is allowed an action that player 1 is not, such an action will be player 2's NE, and player 1's BR will add that action. If player 2 is not allowed an action unavailable to player 1, player 2's BR will be a new action unavailable to player 1, if one exists. Thus at least one of the players add a new action in every GMP game in parallel, until both players add all actions. □

Size of the restricted game. The number of iterations in each algorithm does not provide the full picture of their performance, since iterations can require vastly different computation times. Intuitively, the restricted game in XDO is much larger than in PSRO when both algorithms have the same population size, because XDO induces an extensive-form restricted game with all discovered actions, while PSRO induces a normal-form restricted game with population policies as actions. However, as both algorithms progress, the XDO restricted game is bounded in size by the original game, while PSRO can induce a game with exponentially many actions.

XDO for sparse-support policies. XDO is useful when the policies in the population do not cover the full original game, because when they do then finding the restricted game meta-NE is as hard as solving the original game. The motivation behind XDO is that, in games where the NE policies are supported by few actions in most infostates, XDO has the potential to quickly find these actions and terminate without expanding the full game.

To analyze this behavior, consider the m-clone GMP game, in which there are mn actions partitioned into n equal classes. The actions of the two players are considered a match (with payoff $n-1$ to player 1) if they belong to the same class. In (k,m)-clone GMP, a chance node selects among k identical m-clone GMP games. The following proposition shows that in (k,m)-clone GMP with n classes, XDO terminates after adding at most $2n$ actions for each player, instead of the full game of kmn actions.

Proposition 5. *In (k,m)-clone GMP with n classes, XDO adds at most $2n$ actions for each player.*

Proof. The proof repeats that of Proposition 4, but considering classes instead of actions, because it does not matter which member of a class is added. Once at least one member of each class is added to the restricted game, the meta-NE has full-game exploitability 0, and XDO terminates. In iterations where a BR for a player does not add a new class, it may add a new action member of an existing class. In total, $2n$ actions may be added for each player. □

PSRO lower bound. Similarly to XDO, PSRO can also outperform the guarantee of Proposition 2 in certain cases. Generically, however, the linear upper bound on XDO established by Proposition 3, $\sum_i |\mathcal{I}_i|$, is also a *lower bound* on the normal-form population size of pure strategies that is needed to support a NE in PSRO. To show this, consider the perturbed k-GMP game, in which the payoffs in each GMP game are slightly modified to induce k distinct NE. The following proposition establishes a linear lower bound for PSRO in perturbed k-GMP games.

Proposition 6. *There exist perturbed k-GMP games with n actions, in which PSRO cannot terminate in fewer than $k(n-1)+1$ iterations.*

Proof. The proof is contained in the supplementary materials. □

Neural Extensive-Form Double Oracle (NXDO)

Neural Extensive-Form Double Oracle (NXDO) extends XDO to large games through deep reinforcement learning (DRL). Instead of using an oracle best response, NXDO instead uses approximate best responses that are trained via any deep reinforcement learning algorithm such as DQN (Mnih et al. 2015) or PPO (Schulman et al. 2017). Instead of representing the restricted game explicitly as the set of allowed actions in every infostate, NXDO *delegates* these actions to population policies. A meta-policy in the restricted game selects, in each infostate, the population policy

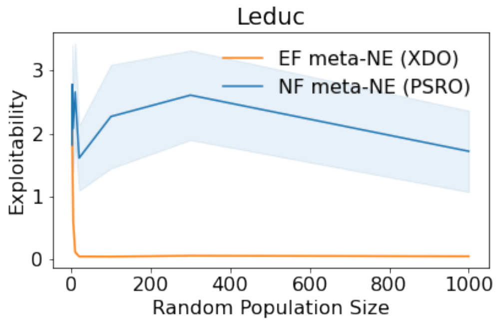

Figure 3

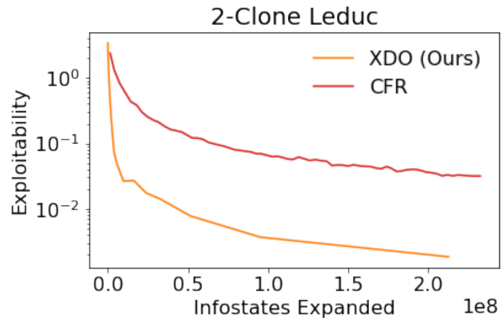

Figure 5

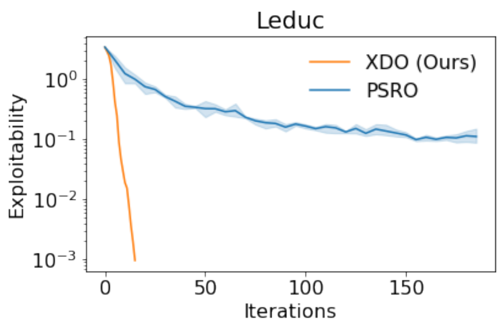

Figure 4

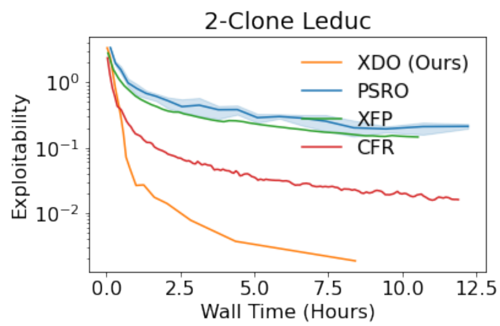

Figure 6

that will be invoked to sample an action. That action is then applied to the original game to get the next state.

Formally, NXDO keeps a population of DRL policies Π^t at time t. Each iteration, a restricted extensive-form game is created and a NE to the restricted game is computed. The restricted game is created by having meta-actions at every infostate that pick one policy from the population.

$$\forall s_i \in \mathcal{I}_i \quad \mathcal{A}_i^r(s_i) = \{1, 2, ..., |\Pi_i^t|\} \tag{2}$$

The world states, observations, and histories remain the same as the original game. After each player selects a meta-action that indicates a population policy, an action is sampled from that population policy and used for the world state transition. The transition function in the restricted game satisfies

$$\mathcal{T}^r(h, a^r, w') = \sum_a \prod_i \pi_i^{a_i^r}(s_i(h), a_i)\mathcal{T}(h, a, w'), \tag{3}$$

where $\pi_i^1, \ldots, \pi_i^{|\Pi_i|}$ are the population policies for player i.

With the restricted game thus defined, an ϵ-NE π^{r*} is computed in this restricted game via a DRL method for finding NE, such as NFSP or DREAM. Approximate BRs $BR_1(\pi_2^{r*})$ and $BR_2(\pi_1^{r*})$ to this meta-NE are computed via a DRL algorithm such as DQN or PPO. These BRs are then added to the population of policies: $\Pi_i^{t+1} = \Pi_i^t \cup BR_i(\pi_{-i}^{r*})$ for $i \in \{1, 2\}$. Provided that the DRL best responses are sufficiently close to oracle best responses and the inner-loop solver finds a sufficiently close approximate NE of the restricted game, NXDO inherits the same convergence properties as XDO. Of course, in practice, contemporary DRL methods lack any guarantee of providing approximate NE or BRs. Nevertheless, we show experimentally that approximate exploitability can decrease through execution of NXDO faster than it does for PSRO and NFSP.

A drawback of meta-actions that delegate actions to population policies is that the number of meta-actions grows linearly with the number of iterations. This can eventually make the restricted game harder to solve than the original game. In our experiments, however, NXDO achieves significant improvements in exploitability within a very small number of iterations, such that the issue of action delegation does not becomes an obstacle.

In games where it is tractable, we consider a variant, NXDO-VA, where the restricted game is explicitly calculated and defined with valid and invalid original-game actions in the same way as with Tabular XDO, using equation restricted$_game$.

Results

For the tabular experiments, we use XDO with an oracle best response (BR) and CFR for the inner-loop meta-NE solver. We compare XDO with PSRO and XFP, which use oracle BRs as well. We also compare with CFR, and for both CFR and XFP we follow the implementations in OpenSpiel (Lanctot et al. 2019). Since CFR, XDO, and XFP are deterministic, we do not plot error bars for these algorithms. For

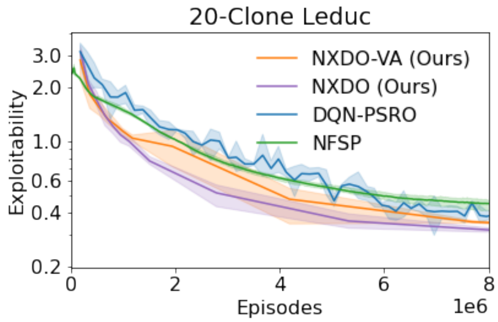

Figure 7

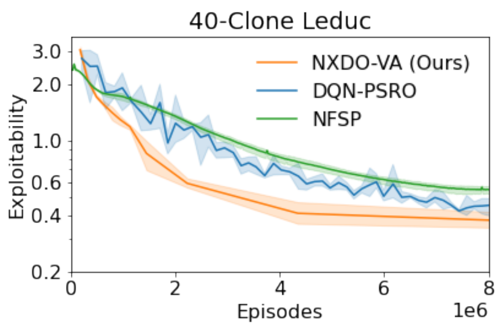

Figure 8

the neural experiments, we use NXDO with a DQN BR and NFSP as the meta-NE solver. DQN-PSRO uses a DQN BR and FP as the meta-NE solver. We compare these algorithms on m-Clone Leduc poker and no-limit poker, described below.

m-Clone Leduc poker: m-Clone Leduc poker is similar to Leduc poker but with every action duplicated m times, such that instead of a single call, fold, and bet action there are m identical call, fold, and bet actions. As the number of cloned actions increases, we expect the performance of methods based on CFR and FP such as DREAM and NFSP to deteriorate, while the performance of XDO remains largely unchanged because it ignores the extra actions.

No-limit poker: We consider a simple no-limit poker game, in which each player is dealt one card from a deck consisting of two suits and three ranks. Then two rounds of betting take place, one pre-flop and one post-flop. During the flop, the dealer deals one public card face up. During each betting round, players can choose to bet an integer amount of chips. Players start out with a stack of 10 chips. The winner of the hand is the player with the highest pair, or the highest ranking card if neither player has a pair.

Comparing Tabular XDO and Oracle PSRO with a fixed population

In extensive-form games, only mixing at the root of the game as done in PSRO can be very inefficient. To demonstrate this, we randomly sample a population of pure strategies and then computed a normal-form meta-NE as in PSRO and an extensive-form meta-NE as in XDO. We randomly sample ten populations of pure strategies of sizes 2, 3, 5, 10, 20, 100, 300, and 1000. After calculating the normal-form and extensive-form meta-NE for a population, we measure its exploitability. In Figure 3 we plot the average final exploitability of both methods for each of the population sizes. The shaded region represents 95% confidence intervals. Even after 1000 random pure strategies, we find that calculating a normal-form meta-NE as in PSRO is still very exploitable. In contrast, the extensive-form meta-NE achieves low exploitability with only 20 random policies.

Comparing Tabular XDO and Oracle PSRO on Leduc poker

As demonstrated by the previous result, finding a normal-form meta-NE can be much less efficient and more exploitable than finding an extensive-form meta-NE. This means that PSRO will usually require many more pure strategies to achieve a similar level of exploitability to XDO. Figure 4 summarizes the results of running XDO and PSRO with an oracle BR. Even after 200 iterations, PSRO remains significantly more exploitable than XDO is at 20 iterations. XDO achieves exploitability of 0.1 in over 20x fewer iterations than PSRO. In large games where calculating many approximate BRs via reinforcement learning is expensive, requiring vastly more iterations can render PSRO infeasible.

Comparing XDO and CFR in 2-Clone Leduc poker by infostates expanded

We compare XDO with CFR in 2-Clone Leduc poker. In Figure 8, we plot the exploitability of these two algorithms as a function of the number of infostates expanded by the algorithm. Since XFP and PSRO only use oracles, we do not include them in this analysis. Since CFR updates every infostate every iteration, as we increase the number of cloned actions, the performance of CFR will deteriorate. In contrast, XDO will tend to not add cloned actions, which allows the inner-loop CFR to expand fewer infostates. These results for XDO are with an oracle best response that always chooses the first available best response. We found that if XDO randomly chose a best response instead, then XDO would still outperform CFR, but not by as much.

Comparing XDO with CFR, PSRO, and XFP in 2-Clone Leduc poker by wall time

We compare XDO with CFR, PSRO, and XFP in 2-Clone Leduc poker by wall time. While not a perfect comparison because each algorithm uses oracles to different extents, it allows us to compare all four algorithms by the same metric. As shown in Figure 6, XDO vastly outperforms the other three methods. Tabular XDO achieves exploitability over

11x lower than CFR and over 82x lower than PSRO and XFP in the same amount of time.

Neural Experiments on 40-Clone Leduc poker

We compare NXDO with NFSP and PSRO on 40-Clone Leduc poker. Similar to the tabular experiments, we find that XDO outperforms both methods. However, we find that the margin by which XDO outperforms these methods is smaller than in tabular experiments. We conjecture that the restricted game induced by XDO is already large enough for the inner-loop NFSP meta-NE solver to struggle. As more adequate NE solvers are developed, in particular CFR-based algorithms, we expect XDO to be able to leverage them as meta-NE solvers to achieve better performance.

Neural Experiments on no limit poker

Table 1: Average Reward

	XDO	PSRO	NFSP
XDO	-	0.615	-0.183
PSRO	-0.615	-	-0.462
NFSP	0.183	0.462	-

Table 2: Average Win Rate

	XDO	PSRO	NFSP
XDO	-	0.512	0.598
PSRO	0.488	-	0.583
NFSP	0.402	0.417	-

Discussion

In this paper, we propose a XDO, a double oracle algorithm that operates directly on the extensive-form of a two-player zero-sum game. XDO mixes among pure strategy best responses at every infostate instead of only at the root of the game as in PSRO. Because of this, only a number of pure best responses that is linear in the number of infostates is needed to find a NE. In contrast, the lowest known upper bound on the number of iterations needed in PSRO is exponential in the number of infostates. To find a meta-NE when mixing at every infostate, tabular XDO runs a tabular algorithm such as CFR on the restricted game defined by only allowing actions that at least one BR in the population chooses at that infostate. We also introduce NXDO, which uses DRL to find approximate BRs and to solve the restricted game.

Our experimental results suggest that XDO significantly outperforms CFR, PSRO, and XFP on 2-Clone Leduc poker and that XDO can find significantly less exploitable meta-NE strategies than PSRO. Other experiments indicate that NXDO significantly outperforms PSRO and NFSP on 40-Clone Leduc poker and outperforms PSRO on a small no-limit poker game.

We conjecture that XDO and NXDO perform well in games where the NE must mix over many infostates, but only a small fraction of all actions are in the support of the NE at each infostate. In such games, we expect XDO and NXDO to outperform PSRO, because PSRO may require a superlinear, or even exponential number of pure strategies. We also expect XDO and NXDO to outperform CFR and NFSP, respectively, on games where the NE only needs to mix over a small number of actions. This is because CFR and NFSP scale poorly with the number of actions in the game, but XDO and NXDO tend to discover a set of relevant actions and to not consider actions that are dominated or redundant. We hypothesize that games having this property are prevalent across a number of domains such as large board games and video games and in robotics applications. We also think that XDO and NXDO might be a promising approach toward solving large continuous-action games.

References

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2017. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748* .

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* .

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

Brown, N.; Kroer, C.; and Sandholm, T. 2017. Dynamic thresholding and pruning for regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

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

Brown, N.; and Sandholm, T. 2015. Regret-Based Pruning in Extensive-Form Games. In *NIPS*, 1972–1980.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Gruslys, A.; Lanctot, M.; Munos, R.; Timbers, F.; Schmid, M.; Perolat, J.; Morrill, D.; Zambaldi, V.; Lespiau, J.-B.; Schultz, J.; et al. 2020. The Advantage Regret-Matching Actor-Critic. *arXiv preprint arXiv:2008.12234* .

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

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

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A. G.; Beattie, C.; Rabinowitz, N. C.;

Morcos, A. S.; Ruderman, A.; et al. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364(6443): 859–865.

Kuhn, H. W.; and Tucker, A. W. 1953. *Contributions to the Theory of Games*, volume 2. Princeton University Press.

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

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, 4190–4203.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, 6379–6390.

Majumdar, S.; Khadka, S.; Miret, S.; Mcaleer, S.; and Tumer, K. 2020. Evolutionary Reinforcement Learning for Sample-Efficient Multiagent Coordination. In *International Conference on Machine Learning*, 6651–6660. PMLR.

McAleer, S.; Lanier, J.; Fox, R.; and Baldi, P. 2020. Pipeline PSRO: A Scalable Approach for Finding Approximate Nash Equilibria in Large Games .

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

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature* 518(7540): 529–533.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485* .

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Steinberger, E.; Lerer, A.; and Brown, N. 2020. DREAM: Deep Regret minimization with Advantage baselines and Model-free learning. *arXiv preprint arXiv:2006.10410* .

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

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, 1729–1736.