Efficient Double Oracle for Extensive-Form Two-Player Zero-Sum Games

Yihong Huang[1,2], Liansheng Zhuang[1(✉)], Cheng Zhao[1], and Haonan Liu[1]

[1] University of Science and Technology of China, Hefei 230027, China
hyh1109@mail.ustc.edu.cn, lszhuang@ustc.edu.cn
[2] Peng Cheng Laboratory, Shenzhen 518000, China

Abstract. Policy Space Response Oracles (PSRO) is a powerful tool for large two-player zero-sum games, which is based on the tabular Double Oracle (DO) method and has achieved state-of-the-art performance. Though having guarantee to converge to a Nash equilibrium, existing PSRO and its variants suffer from two drawbacks: (1) exponential growth of the number of iterations and (2) serious performance oscillation before convergence. To address these issues, this paper proposes Efficient Double Oracle (EDO), a tabular double oracle algorithm for extensive-form two-player zero-sum games, which is guaranteed to converge linearly in the number of infostates while decreasing exploitability every iteration. To this end, EDO first mixes best responses at every infostate so that it can make full use of current policy population and significantly reduce the number of iterations. Moreover, EDO finds the restricted policy for each player that minimizes its exploitability against an unrestricted opponent. Finally, we introduce Neural EDO (NEDO) to scale up EDO to large games, where the best response and the meta-NE are learned through deep reinforcement learning. Experiments on Leduc Poker and Kuhn Poker show that EDO achieves a lower exploitability than PSRO and XFP with the same amount of computation. We also find that NEDO outperforms PSRO and NXDO empirically on Leduc Poker and different versions of Tic Tac Toe.

Keywords: Two-player zero-sum games · Nash equilibrium · Deep reinforcement learning

1 Introduction

Two-player zero-sum games have been a long-standing interest of the development of artificial intelligence. In solving such games, an agent aims to minimize its exploitability, the performance of its opponent in the worst case. When both agents achieve zero exploitability, they reach a Nash equilibrium (NE) [13], a classical solution concept from Game Theory. Even though NE is a clear objective, developing a general algorithm capable of finding an approximate NE often requires tremendous human efforts.

M. Tanveer et al. (Eds.): ICONIP 2022, LNCS 13624, pp. 414–424, 2023.
https://doi.org/10.1007/978-3-031-30108-7_35

Policy Space Response Oracles (PSRO) [6] is a general multi-agent reinforcement learning algorithm which has been applied in many non-trivial tasks. Generally, PSRO aims to find an approximate NE by iteratively expanding a restricted game of a restricted policy population, which is ideally much smaller than the original game. Methods based on PSRO have achieved state-of-the-art performance on large two-player zero-sum games such as Starcraft [16] and Stratego [8]. Despite the empirical success achieved, PSRO and its variants still suffer from exponential growth of the number of iterations and potential serious performance oscillation. The reason for this is that PSRO is based on the tabular method Double Oracle (DO) [11], which makes an inefficient use of the policy population and has no guarantee to decrease exploitability from one iteration to the next.

In this work, we propose a new double oracle algorithm, Efficient Double Oracle (EDO), which is designed for extensive-form two-player zero-sum games. Like PSRO, each player in EDO maintains a population of pure strategies. However, EDO makes full use of the policy population by mixing best responses at every infostate instead of mixing them only at the root of the game, which is what DO does. EDO also removes the restriction on the opponent policy space and creates respective meta-games for each player. EDO is guaranteed to converge to a 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. EDO is also guaranteed to find the least-exploitable policy for the current policy population in each iteration, which avoids the problem of performance oscillation in PSRO.

We also introduce a neural version of EDO, called Neural EDO (NEDO). NEDO uses deep reinforcement learning (DRL) methods to compute a meta-NE in restricted games and compute best responses in each iteration for large games. The restricted games of NEDO contain meta-actions, each selects a corresponding population policy to play the next action. The meta-solver could be any neural extensive-form game solver, such as NFSP [4] and DREAM [15]. NEDO scales up EDO to large games using deep reinforcement learning.

To summarize, our contributions are as follows:

1. We present EDO, a tabular double oracle algorithm that converges linearly without performance oscillation.
2. We present NEDO, a deep reinforcement learning version of EDO that scales up EDO to large games and outperforms existing PSRO methods in all of our experiments.

2 Background

We consider extensive-form games with perfect recall. 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, the space of joint actions for players is denoted as $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$. $\mathcal{A}_i(w) \subseteq \mathcal{A}_i$ denotes the legal action set for player $i \in \{1, \ldots, N\}$ at world state w and $a = (a_1, \ldots, a_N) \in \mathcal{A}$ denotes a joint action. At each world state, a transition function determines the probabilities of the next

world state w' after a joint action a is chosen, which is denoted as $\mathcal{T}(w, a) \in \Delta^{\mathcal{W}}$. Player i makes an *observation* $o_i = \mathcal{O}_i(w, a, w')$ upon each transition. The game ends when the players reach a terminal world state w^T and player i receives a reward $\mathcal{R}_i(w)$ in each world state w.

A *history* is a sequence of actions and world states representing a trace of the game, denoted $h = (w^0, a^0, w^1, a^1 \ldots, w^t)$, where w^0 is the initial world state of the game. $\mathcal{R}_i(h)$ and $\mathcal{A}_i(h)$ are the reward and the set of legal actions for player i in the last world state of h. An *infostate* for player i, denoted by s_i, is a sequence of that player's observations and actions until that time, denoted $s_i(h) = (a_i^0, o_i^1, a_i^1, \ldots, o_i^t)$. It's also assumed that different world states corresponding to the same infostate share the same set of legal actions $\mathcal{A}_i(s_i(h)) = \mathcal{A}_i(h)$.

A player's *strategy* π_i is a function mapping from an infostate to a distribution over legal actions. A *strategy profile* π is a tuple $(\pi_1, \ldots, \pi_N)$. All players except i are denoted $-i$ and their joint strategies are denoted π_{-i}. A strategy for a history is defined as $\pi_i(h) = \pi_i(s_i(h))$. We also refer to a strategy as a *policy* in the later parts of this article.

The *value* $v_i^{\pi}(h)$ is the expected sum of future rewards given that all players play the strategy profile π. The value of the entire game is denoted as $v_i(\pi)$. A *two-player zero-sum* game has $v_1(\pi) + v_2(\pi) = 0$ for all strategy profiles π. A *Nash equilibrium (NE)* is a strategy profile π^* satisfying $v_i(\pi^*) = \max_{\pi_i}(\pi_i, \pi^*_{-i})$ for each player i.

A *best response* to π_{-i} is defined as $\mathbb{BR}_i(\pi_{-i}) = \arg\max_{\pi_i} v_i(\pi_i, \pi_{-i})$ and the *exploitability* of a strategy profile π is defined as $e(\pi) = \sum_{i \in \mathcal{N}} \max_{\pi'_i} v_i(\pi'_i, \pi_{-i})$. Lower exploitability means better approximation to Nash equilibrium for a strategy profile.

A *normal-form game* is a single-step extensive-form game, in which players choose their actions simultaneously. An extensive-form game can always induce a normal-form game in which legal actions for player i are its deterministic strategies in the original game.

3 Related Work

In small two-player zero-sum games, Nash equilibrium can be found via linear programming and no-regret algorithms such as replicator dynamics, fictitious play (FP) and regret matching, which become infeasible when the size of game increases. In perfect information extensive-form games, algorithms based on minimax tree search have had success on games such as chess and Go [14]. Extensive-form fictitious play (XFP) [3] and counterfactual regret minimization (CFR) [18] extend FP and regret matching, respectively, to extensive-form games. Although CFR based on abstraction has been used in large imperfect-information extensive-form zero-sum games like heads-up no limit Texas Hold'em [2], this is not a general method for large games because finding efficient abstractions needs expert domain knowledge.

Recently, deep reinforcement learning (DRL) [7] has been proven effective on complex sequential decision-making problems like Atari games. Deep CFR [1] is a

general method that trains a neural network on a buffer of counterfactual values. However, external sampling used by Deep CFR may be impractical for games with a large branching factor. AlphaStar based on self-play beat top human players at StarCraft using population-based reinforcement learning. Nevertheless, self-play methods are not guaranteed to converge to an approximate NE and can not handle some small games like Rock-Paper-Scissors. Neural Fictitious Self Play (NFSP) approximates XFP by progressively training a best response against an average of all past opponent policies via DRL. The average policy in NFSP is also represented by a neural network and trained by supervised learning using a reservoir-sampling buffer, which may become prohibitively large in large games.

Double Oracle (DO) is an algorithm for finding a NE in normal-form games. The algorithm works by maintaining a population of policies Π^t at time t. A meta-Nash Equilibrium (meta-NE) is computed for the game restricted to policies in Π^t in each iteration. Then, a best response to the meta-NE for each player is computed and added to the population. In the worst case, DO must expand all pure strategies. Policy Space Response Oracles (PSRO) approximates the DO algorithm. The meta-NE in PSRO is computed on the empirical game matrix, which is generated by sampling each pair of policies and tracking the average utility. The approximate best response in PSRO is computed via any reinforcement learning method. Extensive-form Double Oracle (XDO) [9] and its neural version, Neural XDO (NXDO) iteratively adds extensive-form BRs to a population and then computes a meta-NE on an entensive-form meta-game while this method is not guaranteed to decrease exploitability every iteration, which means potential performance oscillation.

The concept of finding a low-exploitability meta-policy in a restricted game has also been explored in recent years [10,17]. However, most of these work focus on normal-form restricted games. This paper proposes a method to compute the least-exploitable meta-policy in extensive-form restricted games.

4 Method

4.1 Efficient Double Oracle

In this paper, we propose Efficient Double Oracle (EDO), which is guaranteed to converge linearly and decrease the exploitability of meta-NE monotonically. Like other DO algorithms, EDO maintains a population of pure strategies and computes a meta-NE of a restricted game in each iteration. Also like other DO algorithms, EDO expands its population by best responses (BR) to the meta-NE. However, to overcome aforementioned drawbacks of PSRO, EDO creates different extensive-form restricted games instead of traditional normal-form ones for each player, and the restricted game G^i for player i is constructed while only this player is restricted to play actions suggested by any strategy in the population at each infostate, against the opponent able to play all legal actions in the full game.

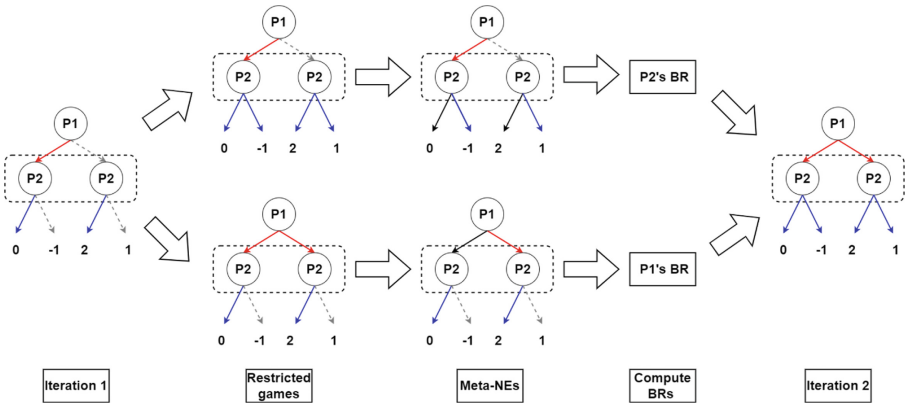

Fig. 1. In EDO, the following happens in one iteration: (1)Two extensive-form restricted games are created for each player. (2)For each restricted game, a meta-NE is computed. (3)For each player, a BR is trained against the opponent's restricted policy and this BR is added to the population. Dashed actions are outside the restricted games and the black solid actions are outside the meta-NE.

Formally, EDO creates a restricted extensive-form game for each player based on a pure strategy population Π^t and computes a restricted NE at time t. Each restricted game G^p for player p has the same infostates as the full game with restricted actions:

$$\mathcal{A}_i^p(s_i) = \begin{cases} \left\{a \in \mathcal{A}_i(s_i) : \exists \pi_i \in \Pi_i^t \text{ s.t. } \pi_i(s_i, a) > 0\right\}, & i = p, \\ \mathcal{A}_i(s_i), & i \neq p. \end{cases} \tag{1}$$

The restricted game G^i for player i is then solved for both players to get a meta-NE (π_1^i, π_2^i). EDO uses a tabular method like XFP or CFR as the restricted game solver. We refer to the restricted player's strategy as the restricted NE $\pi_i^r = \pi_i^i$. The restricted NE π^r is the strategy profile with the least exploitability supported by current population. Then, a novel best response with some unseen action against the meta-NE of the restricted opponent is computed and added to the strategy population for each player. Therefore, at each non-final iteration of EDO, at least one new action at some non-terminal infostate is added to the extensive-form restricted game by some player, otherwise the algorithm terminates. As a result, the upper bound of the number of iterations is linear to the number of infostates while the number of iterations DO takes to terminate is exponential to the number of infostates in the worst case, which may make computing meta-NE intractable in later iterations. To illustrate how EDO works, we demonstrate a simple game in Fig. 1. Note that in large games this restricted game may become prohibitively large to solve, which requires NEDO, introduced later in this paper. EDO is described in Algorithm 1.

Algorithm 1: EDO

Input: Initial Population Π^0, $t = 0$
repeat
  Define restricted game G^i for Π^t via eq. 1, for $i \in \{1, 2\}$
  $\pi_i^r \leftarrow$ NE policy of player i in G^i
  for $i \in \{1, 2\}$ **do**
    Compute a novel best response $\beta_i \leftarrow \mathbb{BR}_i(\pi^r_{-i})$
    $\Pi_i^{t+1} = \Pi_i^t \cup \beta_i$
  end
  $t = t + 1$
until No action of best responses is outside G^i;
return π^r

EDO is guaranteed to terminate because the number of actions of every infostate in a extensive-form game is finite. The returned retricted NE π^r is also a NE in the original game (Proposition 1), as shown below.

Proposition 1. *The restricted NE of both players is a Nash equilibrium in the original game, when EDO terminates.*

Proof. Suppose that (π_1^r, π_2^{ur}) and (π_1^{ur}, π_2^r) are the meta-NE of the restricted games for player 1 and 2, respectively, when the algorithm is terminated. Let β_1 and β_2 be the novel best responses to π_2^r and π_1^r. If β_1 or β_2 on some infostate has support outside the respective restricted game, EDO would not terminate. Therefore, the support of β_1 and β_2 on all infostates is guaranteed to be inside the respective restricted game, which means π_1^{ur} and π_2^{ur} also have this property. Refer to the policy space of the restricted players as Λ^r. Then:

$$\begin{aligned} v_2(\pi_1^r, \pi_2^r) &\leq v_2(\pi_1^r, \pi_2^{ur}) \\ &= \min_{\pi_1 \in \Lambda_1^r} v_2(\pi_1, \pi_2^{ur}) \\ &\leq v_2(\pi_1^{ur}, \pi_2^{ur}) \\ &\leq \max_{\pi_2 \in \Lambda_2^r} v_2(\pi_1^{ur}, \pi_2) \\ &= v_2(\pi_1^{ur}, \pi_2^r) \\ &= \min_{\pi_1} v_2(\pi_1, \pi_2^r) \\ &\leq v_2(\pi_1^r, \pi_2^r). \end{aligned} \tag{2}$$

$$v_2(\pi_1^r, \pi_2^r) = v_2(\pi_1^r, \pi_2^{ur}) = \max_{\pi_2} v_2(\pi_1^r, \pi_2). \tag{3}$$

Therefore, player 2 has no motivation to deviate from π_2^r, which is same for player 1. Hence, π^r is a NE in the original game.

While DO and EDO both return a NE upon termination, EDO has the guarantee of non-increasing monotonicity on exploitability (Propostition 2), which doesn't hold for DO.

Proposition 2. *The exploitability of EDO is monotonically non-increasing.*

Proof. Suppose that π^t is the meta-NE at iteration t, and Λ_i^t is the policy space of player i in the restricted game G^i. It's obvious that $\Lambda_i^t \subseteq \Lambda_i^{t+1}$ because $\Pi_i^t \subseteq \Pi_i^{t+1}$. Then,

$$\begin{aligned} e_i(\pi_i^t) &= -\min_{\pi_{-i}} v_i(\pi_i^t, \pi_{-i}) \\ &= -\max_{\pi_i \in \Lambda_i^t} \min_{\pi_{-i}} v_i(\pi_i, \pi_{-i}) \\ &\geq -\max_{\pi_i \in \Lambda_i^{t+1}} \min_{\pi_{-i}} v_i(\pi_i, \pi_{-i}) \\ &= e_i(\pi_i^{t+1}). \end{aligned} \tag{4}$$

$$e(\pi^t) = \sum_i e_i(\pi_i^t) \geq \sum_i e_i(\pi_i^{t+1}) = e(\pi^{t+1}). \tag{5}$$

4.2 Neural Efficient Double Oracle

As mentioned above, the tabular EDO cannot handle large games because the restricted games may become prohibitively large to solve. Hence, we propose Neural Efficient Double Oracle (NEDO) algorithm, which extends EDO to large games by deep reinforcement learning (DRL) methods. NEDO uses approximate best responses trained by DRL and also uses DRL methods as the meta-solver. However, approximate best responses usually have many actions with positive probability at every infostate while a oracle best response usually has a single action with probability 1, which means restricted action space defined in EDO may be quite large and even equal to the original game with approximate BRs. To avoid this problem, NEDO defines a different meta-action space for the restricted game G^p for each player p based on the player's DRL policy population Π_p:

$$\mathcal{A}_i^p(s_i) = \begin{cases} \{1, 2, ..., |\Pi_i|\}, & i = p, \\ \mathcal{A}_i(s_i), & i \neq p. \end{cases} \tag{6}$$

NEDO is described in Algorithm 2. Although the action space differs, the restricted game states, histories and infostates are still the same as in the original game. After the restricted player p chooses an action indicating a DRL population policy, it would actually sample an action in the orginal game according to this DRL policy. The transition function of G^p then becomes:

$$\mathcal{T}^p(h, a^p, w) = \sum_a \pi_p^{a_p^p}(s_p(h), a_p) \mathbb{1}(a_{-p}^p = a_{-p}) \mathcal{T}(h, a, w). \tag{7}$$

After the restricted game for each player i is defined, an approximate meta-NE (π_1^{i*}, π_2^{i*}) is computed via a DRL method like NFSP. We refer to the restricted player's strategy as the restricted NE $\pi_i^{r*} = \pi_i^{i*}$. Approximate BRs to π_2^{r*} and π_1^{r*} are computed via a DRL method like DQN [12] and added to the corresponding policy polulation: $\Pi_i^{t+1} = \Pi_i^t \cup \mathbb{BR}_i(\pi_{-i}^{r*}),\ i \in \{1, 2\}$.

Algorithm 2: NEDO

Input: Initial Population Π^0, $t = 0$
repeat
  Define restricted game G^i for Π^t via eq. 2, for $i \in \{1, 2\}$
  $\pi_i^{r*} \leftarrow$ approximate NE policy of player i in G^i via DRL
  for $i \in \{1, 2\}$ **do**
    Compute an approximate best response $\beta_i \leftarrow \mathbb{BR}_i(\pi_{-i}^{r*})$ via DRL
    $\Pi_i^{t+1} = \Pi_i^t \cup \beta_i$
  end
  $t = t + 1$
until *termination conditions are satisfied*;
return π^{r*}

Although contemporary DRL methods lack guarantee of the approximate solutions, we show that NEDO outperforms PSRO and some other methods experimentally in later sections. A potential drawback of NEDO is that the meta-action space of the restricted player in the corresponding restricted game grows linearly with the number of iterations, which may cause the restricted game hard to solve in later iterations, even harder than the original game. Nevertheless, it's shown that the algorithm achieves significant performance with a small number of iterations in our experiments, which means this issue does not become an real obstacle.

5 Experiments

In this section, we report experiments upon EDO and NEDO. For the tabular experiments, we use EDO with an oracle best response and CFR as the meta-solver. For the neural experiments with deep reinforcement learning methods, we use NEDO taking DQN as the best response solver and NFSP as the meta-solver. In both experiments, we use some extensive-form zero-sum games in Openspiel [5] as our environments and report results of different algorithms on each environment respectively.

5.1 Experiments with Tabular Methods

As mentioned above, DO often requires more pure strategies to achieve an approximate NE of the original game and the exploitability is not guaranteed to decrease every iteration. Figure 2(a) presents a bad case of DO, in which DO increases exploitability every iteration except the last one, while EDO always decreases it. The game is the equivalent extensive-form game of a zero-sum normal-form game, in which all values are 0, except if the row r is one more than the column c or the column c is one more than the column r. The values of these are $\sum_{i=0}^{r-1} 4^i + 3i$ and $\sum_{i=0}^{c-1} -4^i + 3i$ respectively. We plot the performance of EDO and DO in this game with 10 actions.

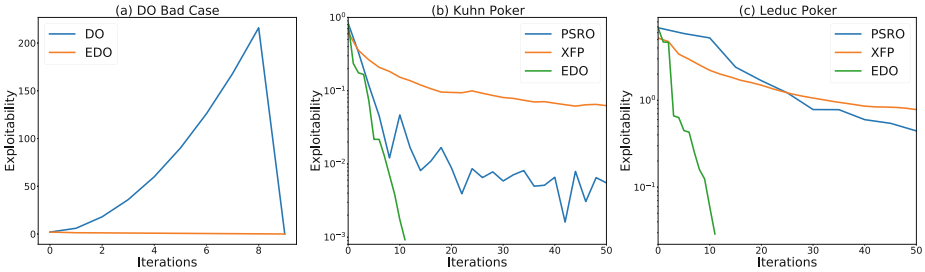

Fig. 2. (a) DO Bad Case; (b) Exploitability in Kuhn Poker of PSRO, XFP and EDO; (c) Exploitability in Leduc Poker of PSRO, XFP and EDO.

We also compare EDO with XFP and PSRO in Kuhn Poker and Leduc Poker in Fig. 2(b) and Fig. 2(c). XFP and PSRO here also use oracle BRs and PSRO uses fictitious play (FP) as the meta-solver. We plot the exploitability of these algorithms as a function of iterations. It's shown that EDO achieves lower exploitability in much fewer iterations than PSRO and XFP. Although EDO requires more computation in a single iteration because the policy space of an extensive-form restricted game is often larger than the one of a normal-form restricted game for the same population, much less iterations to achieve a low exploitability means less amount of computation in total. In larger games, the expensive cost of computing BRs so many times would also make PSRO infeasible.

5.2 Experiments with Deep Reinforcement Learning

For all neural expeiments, we use the same DRL best response hyperparameters in NEDO, PSRO and NXDO as well as in the measure of approximate exploitability. In DQN, the learning rate of DQN is 0.01 and the size of replay buffer is 2e5. In NFSP, the anticipatory factor is set to be 0.1 and the learning rate of average network in NFSP is 0.1. For PSRO, the entries of empirical payoff tables are computed by sampling 1000 episodes for each new pair of populations strategies. FP is used as the meta-NE solver in PSRO and the number of inner-loop iterations for FP is 2000.

In Fig. 3(a), we compare the exploitability of NEDO and PSRO and NXDO on Leduc Poker. DQN is used to train BRs in these methods. Although NEDO uses DRL methods to solve the restricted games and train BRs, we find that the number of iterations it takes to achieve a low exploitability is much smaller than PSRO and similar to NXDO, which means NEDO inherits the property of EDO in exntensive-form games empirically.

We also compare NEDO, PSRO and NXDO on Tic Tac Toe and Phantom Tic Tac Toe. We plot the exploitability of these methods as a funtion of episodes. However, it's shown that NXDO does not outperform other methods as much as tabular EDO. One of these reasons could be training episodes of NFSP takes a large proportion compared with training BRs via DQN. Although episodes

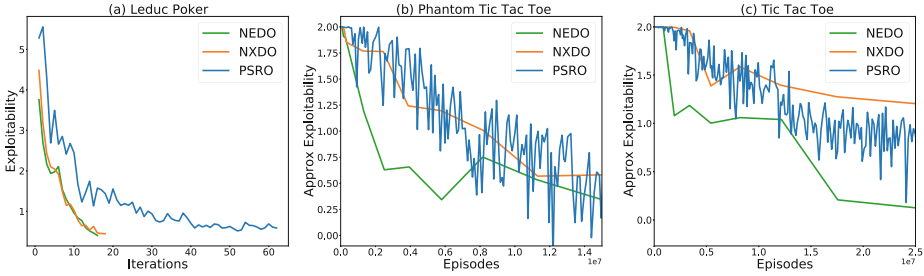

Fig. 3. (a) Exploitability in Leduc of NEDO, NXDO and PSRO as a function of iterations; (b and c) Approximate Exploitability in (Phantom) Tic Tac Toe of NEDO, NXDO and PSRO as a function of episodes gathered.

required by a single iteration in NEDO is more than PSRO, NEDO outperforms PSRO and NXDO in both Tic Tac Toe and Phantom Tic Tac Toe which we conjecture is due to the unrestricted opponent policy space and more effective use of population strategies.

6 Conclusion

In this paper, we propose EDO, a modification of DO that converges to a Nash equilibrium linearly decreasing exploitability monotonically. We also propose NEDO, an algorithm that scales up EDO to large games via deep reinforcement learning. As shown in our tabular experiments, EDO outperforms PSRO and XFP on different poker games. In the neural experiments, it's shown that NEDO outperforms PSRO and NXDO on Leduc Poker and different Tic Tac Toe games, which we conjecture is due to the unrestricted opponent policy space in restricted games and more effective use of population strategies.

Acknowledgements. This work was supported in part to Dr. Liansheng Zhuang by NSFC under contract No. U20B2070 and No. 6197619, and in part to Dr. Houqiang Li by NSFC under contract No. 61836011.

References

1. Brown, N., Lerer, A., Gross, S., Sandholm, T.: Deep counterfactual regret minimization. In: International Conference on Machine Learning, pp. 793–802. PMLR (2019)
2. Brown, N., Sandholm, T.: Superhuman AI for heads-up no-limit poker: libratus beats top professionals. Science **359**(6374), 418–424 (2018)
3. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning, pp. 805–813. PMLR (2015)
4. Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games. arXiv preprint arXiv:1603.01121 (2016)

5. Lanctot, M., et al.: Openspiel: a framework for reinforcement learning in games. arXiv preprint arXiv:1908.09453 (2019)
6. Lanctot, M., et al.: A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
7. Li, Y.: Deep reinforcement learning: an overview. arXiv preprint arXiv:1701.07274 (2017)
8. McAleer, S., Lanier, J.B., Fox, R., Baldi, P.: Pipeline psro: a scalable approach for finding approximate nash equilibria in large games. Adv. Neural Inf. Process. Syst. **33**, 20238–20248 (2020)
9. McAleer, S., Lanier, J.B., Wang, K.A., Baldi, P., Fox, R.: Xdo: a double oracle algorithm for extensive-form games. Adv. Neural Inf. Process. Syst. **34**, 23128–23139 (2021)
10. McAleer, S., Wang, K., Lanctot, M., Lanier, J., Baldi, P., Fox, R.: Anytime optimal psro for two-player zero-sum games. arXiv preprint arXiv:2201.07700 (2022)
11. McMahan, H.B., Gordon, G.J., Blum, A.: Planning in the presence of cost functions controlled by an adversary. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 536–543 (2003)
12. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
13. Nash, J.F., Jr.: Equilibrium points in n-person games. Proc. Natl. Acad. Sci. **36**(1), 48–49 (1950)
14. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
15. Steinberger, E., Lerer, A., Brown, N.: Dream: deep regret minimization with advantage baselines and model-free learning. arXiv preprint arXiv:2006.10410 (2020)
16. Vinyals, O., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)
17. Wang, Y., Ma, Q., Wellman, M.P.: Evaluating strategy exploration in empirical game-theoretic analysis. arXiv preprint arXiv:2105.10423 (2021)
18. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Advances in Neural Information Processing Systems, vol. 20 (2007)